NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

XML BASED ADAPTIVE IPSEC POLICY MANAGEMENT IN A TRUST MANAGEMENT CONTEXT

by

Raj Mohan

September 2002

Thesis Advisor:	Cynthia E. Irvine
Second Reader:	Timothy E. Levin

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2002	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: XML Based Adaptive IPsec Policy Management in a Trust Management Context		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Raj Mohan			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited		**12b. DISTRIBUTION CODE** A	

13.   ABSTRACT (maximum 200 words)

TCP/IP provided the impetus for the growth of the Internet and the IPsec protocol now promises to add to it the desired security strength. IPsec provides users with a mechanism to enforce a range of security services for both confidentiality and integrity, enabling them to securely pass information across networks. Dynamic parameterization of IPsec further enables security mechanisms to adjust the level of security service "on-the-fly" to respond to changing network and operational conditions. The IPsec implementation in OpenBSD works in conjunction with the Trust Management System, KeyNote, to achieve this. However the KeyNote engine requires that an IPsec policy be defined in the KeyNote specification syntax. Defining a security policy in the KeyNote Specification language is, however, extremely difficult and the complexity of the language could lead to incorrect specification of the desired policy, thus degrading the security of the network. This thesis looks into an alternative XML representation of this language and a graphical user interface to evolve a consistent and correct security policy. The interface has the simplicity of a simple menu-driven editor that not only provides KeyNote with a policy in the specified syntax but also integrates techniques for correctness verification and validation.

14. SUBJECT TERMS KeyNote, ISAKMP, IKE, IPsec, Graphical User Interface, Security Assoication (SA), Security Policy Database (SPD), XML, XSLT, DTD, Schema, JDOM, SAX, Security Policy.		**15. NUMBER OF PAGES** 89
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

i

THIS PAGE INTENTIONALLY LEFT BLANK

XML BASED ADAPTIVE IPSEC POLICY MANAGEMENT IN A TRUST MANAGEMENT CONTEXT

Raj Mohan
Major, Indian Army
ME, Indian Institute Of Science, 1998

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE
AND
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2002**

Author: Raj Mohan

Approved by: Cynthia E. Irvine
                 Thesis Advisor

                 Timothy E. Levin
                 Second Reader/Co-Advisor

                 Christopher S. Eagle
                 Chairman, Department of Computer Science

                 Dan C Boger
                 Chairman, Department of Information Technology Management

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

TCP/IP provided the impetus for the growth of the Internet and the IPsec protocol now promises to add to it the desired security strength. IPsec provides users with a mechanism to enforce a range of security services for both confidentiality and integrity, enabling them to securely pass information across networks. Dynamic parameterization of IPsec further enables security mechanisms to adjust the level of security service "on-the-fly" to respond to changing network and operational conditions. The IPsec implementation in OpenBSD works in conjunction with the Trust Management System, KeyNote, to achieve this. However the KeyNote engine requires that an IPsec policy be defined in the KeyNote specification syntax. Defining a security policy in the KeyNote Specification language is, however, extremely difficult and the complexity of the language could lead to incorrect specification of the desired policy, thus degrading the security of the network. This thesis looks into an alternative XML representation of this language and a graphical user interface to evolve a consistent and correct security policy. The interface has the simplicity of a simple menu-driven editor that not only provides KeyNote with a policy in the specified syntax but also integrates techniques for correctness verification and validation.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

THIS PAGE INTENTIONALLY LEFT BLANK

I.   INTRODUCTION

A.   PURPOSE

Network Protocols such as IPsec and trust management systems like Keynote provide mechanisms to secure computer-to-computer communications. These tools enable the user to use various encryption and authentication mechanisms to ensure confidentiality, integrity and non-repudiation of communications. The trust management system, Keynote specifies a language for describing actions, which are operations with security consequences that are to be controlled by the system. The language also provides the syntax for specifying the application policies, which govern the actions that the principals[1] are authorized to perform. To translate a desired organizational security policy into the Keynote specification language is however a daunting task due to the technical complexity of the language. An incorrect specification of the security policy might result in compromising the network security and is unacceptable. It is in this context that the need for an alternative policy specification mechanism is felt. This mechanism should enable the user to correctly specify the policy and also verify that the specified policy is free of inconsistencies and contradictions. The purpose of this thesis is to analyze, design and implement a policy editor interface that guides a user to specify various attributes of the IPsec security policy. The program will automatically generate the equivalent policy in the Keynote specification language. Alternate presentation mechanisms will be studied to provide the user with an intuitive presentation and to prevent inconsistencies and contradictions in the specified policy.

B.   BACKGROUND

The increased dependence on computers for communication has enhanced the importance of network security. The use of the inherently insecure Internet as the medium for communicating sensitive material requires that the end users have capabilities to ensure that the data transmitted is secure. Furthermore, network administrators should have means to translate the desired organizational security policy

---

[1] A Principal is an entity that is either the authorizer of an assertion or the target of such an assertion. A Principal may be an arbitrary string or a cryptographic key that can be used to sign assertions.

into an automated security policy and have mechanisms to implement this policy over their network.

IPsec extends the IP Protocol to enable security for TCP/IP communications. IPsec provides both secrecy and integrity services. A wide variety of choices are available when establishing protected communications across the network. The appropriate choice and combination of secrecy and integrity mechanisms will depend upon the "trust relationships" between the communicating entities. Those relationships are constrained by the policy of each entity. Negotiation of policy and mechanisms takes place in the context of the Internet Key Exchange (IKE) framework and the Internet Security Association and Key Management Protocol (ISAKMP) (Maughan, Schertler, Turner, Schneider, 1998). However, IKE and ISAKMPD do not provide a general mechanism for managing and incorporating security policy. In order to ensure that IPSEC consistently meets the local security policy needs of the user, a Trust Management System is used to encode policy and support communications security negotiation and management. (Thayer, Doraswamy and Glenn1998)

A trust management system unifies the elements of security policy, credentials, access control, and authorization. Applications can use the Keynote trust management system to verify, through the compliance checker, whether a requested policy addition or change is authorized. (Blaze, Ioannidis, and Keromytis, Feb 2000)

IPsec implementation in open BSD utilizes a trust management system to manage security according to policy. Quality of Security Service (QoSS) provides a means to manage security services based on the requirements set by the user's requests, the system's security policy, the availability of system resources and the network environment. (Irvine and Levin, September 2000)

Dynamic parameterization of IPsec (Agar, December 2001) provides more granularity in IPsec and provides flexibility to adjust security controls according to changes in threat conditions, critical time transmissions, and network congestion/traffic. This makes IPsec a QoSS mechanism.

All the above mechanisms depend on having in place a correct security policy specified in the Keynote specification language. For any practical real life network

2

operations specifying such a dynamic and granular policy is an insurmountable task due to the syntactic complexity of the KeyNote language and the inherent complexity of the policy logic involved. An XML-based specification of the policy should provide the desired flexibility, be easy to use and provide an interface for administration of the security policy. This would provide an abstraction to the KeyNote language and enable users to derive the power of IPsec and KeyNote in managing network security.

C.   EXPECTED BENEFITS OF THE RESEARCH

By providing a policy management toolkit it would be possible to unleash the power of IPsec usage and would enable government and military security systems to automate security service adjustments according to dynamic environmental parameter settings, such as INFOCON and THREATCON. The use of XML in such an effort will enable us to use all the available XML tools for ensuring consistency and also utilize the flexibility and compatibility that XML provides. The power of XML security can also be harnessed to enhance the overall security of the communicating systems. An easy to use interface will ensure its use and the correctness will give the desired confidence in the overall security implementation of the network. The exploration of XML will also open doors to further research in other areas of computer security that will benefit from it.

D.   RESEARCH OBJECTIVES

The objectives of this research are three fold:

o   Study the Keynote trust management system and IPsec, and explore available XML technologies.

o   Design and develop a policy editor interface to capture the security policy requirements of a user, check for inconsistencies and transform the stated policy into the Keynote specification language.

o   Design and develop a method of incorporating the use of XML to enhance the flexibility, maintainability and interoperability of the policy specifications.

3

E.   LAYOUT OF THE THESIS

The thesis will be organized as follows:

o   Chapter II   Previous Work – This chapter consists of a brief survey of related research.

o   Chapter III XML and Keynote policy specification language – a review of Keynote language and its specification for the QOSS implementation in OpenBSD 2.8. Relevant XML technologies and its application to the problem domain will be reviewed.

o   Chapter IV Design and Implementation – the design philosophy of the toolkit, the considerations and overall architecture will be discussed in detail. Implementation issues of the components will be highlighted in this chapter.

o   Chapter V Research Summary and Future Work – This chapter will summarize the research done and will end with a discussion of future work.

II.   PREVIOUS WORK

A.   IPSEC

The popularity of the Transfer Control Protocol/ Internet Protocol (TCP/IP) and the growing use of computer networks for governmental and business made these protocols vulnerable to scrutiny, attacks and misuse. TCP/IP, which was designed to provide packet based communications over unreliable telephone networks, was not designed for providing secure communications. The first attempt to provide security involved a simple "protect-all" approach to network security i.e. the Virtual Private Networks (VPN). IPsec was then developed to address security vulnerabilities inherent in the Internet Protocol (IP), by defining a more flexible security mechanism for sending data across an insecure medium. IPsec introduced the ability to provide a range of security services ultimately defined by a security policy (See Figure 1). The security policy defines specific security services for each packet, according to packet characteristics such as source and destination addresses.
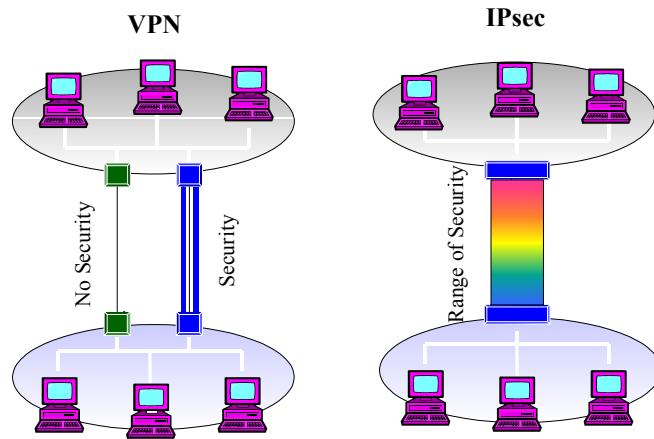
Figure 1.   VPN vs. IPsec Security Mechanisms
(From: Agar, Chris December 2001)

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put

in place any cryptographic keys required to support the requested services. There are two extension headers that follow the main IP header and which incorporate the security features of IPsec. The extension header for authentication is known as the Authentication Header (AH) and that for encryption is known as the Encapsulating Security Payload (ESP) header. The difference between the AH and the authentication within ESP is essentially the amount of content of the packet and headers that is authenticated. Figure 2 and Figure 3 depict the coverage of ESP and AH respectively.

ESP- Protected IP Packet
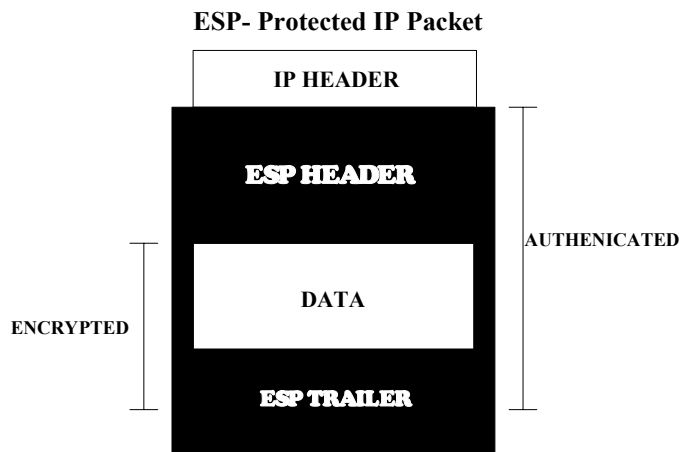
Figure 2.　　ESP- Protected IP Packet.
(After: Doraswamy, Naganand and Harkins, Dan, 1999, 49)

AH- Protected IP Packet
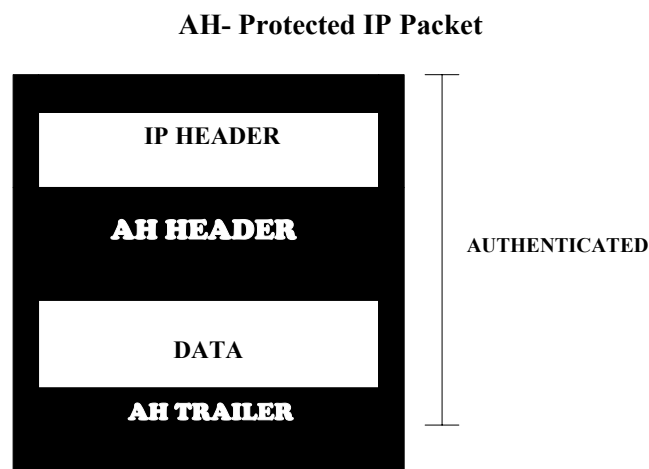
Figure 3.　　AH-Protected IP Packet.
(After: Doraswamy, Naganand and Harkins, Dan, 1999, 51)

IPsec was designed to provide an efficient and effective cryptographic security mechanism for IP version 4 and IP version 6. The mechanism provides the following services: access control, connectionless integrity, data origin authentication, protection against replays,(a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality. These services are applied at the IP layer, providing security for IP and/or upper layer protocols. (Kent, S and Atkinson, R, 1998) The cryptographic algorithms are applied in accordance with system security policies that are defined within IPsec. IPsec can be used on a variety of system architecture models: host-to-host, gateway-to-gateway and gateway-to-host/host-to-gateway. (Doraswamy, Naganand and Harkins, Dan, 1999, 57-79).

1. **Security Associations**

A key concept that appears in both the authentication and confidentiality mechanisms for IP is the Security Association (SA). An association is a one-way relationship between the sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed for two-way secure exchange, then two security associations are required. A Security Association is uniquely defined by three parameters:

o Security Parameters Index (SPI),

o IP destination address, and

o Security protocol identifier.

2. **SA Selectors**

IPsec provides the user with considerable flexibility in the way in which IPsec services are applied to IP traffic. SAs can be combined in a number of ways to yield the desired security configuration. Furthermore, IPsec provides a high degree of granularity in discriminating between traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec, in the former case relating IP traffic to specific SAs.

The means by which IP traffic is related to specific SAs (or no SAs in the case of traffic allowed to bypass IPsec) is the nominal Security Policy Database (SPD). Each SPD entry is defined by a set of IP and upper-layer protocol field values, called selectors.

In effect, these selectors are used to filter outgoing and incoming traffic in order to map it into a particular SA. Selectors are of many types for e.g. destination IP address, transport layer protocol, IP Sec protocol (AH or ESP or AH/ESP), source and destination ports etc.

3. Transport Mode and Tunnel Modes

Transport Mode	Tunnel Mode
IP HEADER	TUNNEL IP HEADER
IPSEC HEADER	IPSEC HEADER
TCP/UDP HEADER	IP HEADER
	TCP/UDP HEADER
DATA	DATA
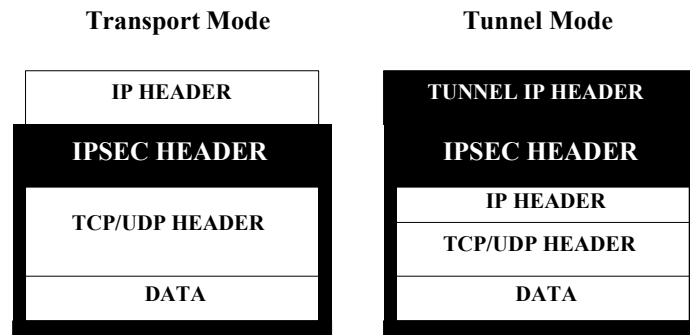
Figure 4.   IPsec Transport and Tunnel Modes.
(After: Doraswamy, Naganand and Harkins, Dan, 1999, 57-80)

Both AH and ESP support two modes of use: transport and tunnel modes depicted in Figure 4 above. Transport mode provides protection primarily for upper layer protocols. Its protection extends to the payload of an IP packet. Typically, transport mode is used for end-to-end communication between two hosts. (e.g., a client and a server, or two workstations). Tunnel mode on the other hand provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new 'outer' IP packet with a new outer IP header. The entire original, or inner, packet travels through a 'tunnel' from one point of an IP network to another. Tunnel mode is most commonly used when one or both ends of an SA is a security gateway, such as a firewall or router that implements IPsec.

4. Combining Security Associations

An individual SA can specify either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for services provided by both AH and ESP.

Further, a particular traffic flow may require IPsec services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPsec services. The term security association bundle refers to a sequence of SAs through which traffic must be processed to provide a desired set of IPsec services. The SAs in a bundle may terminate at different end points or at the same endpoints. Security associations can be combined into bundles in two ways:

a. *Transport Adjacency:*
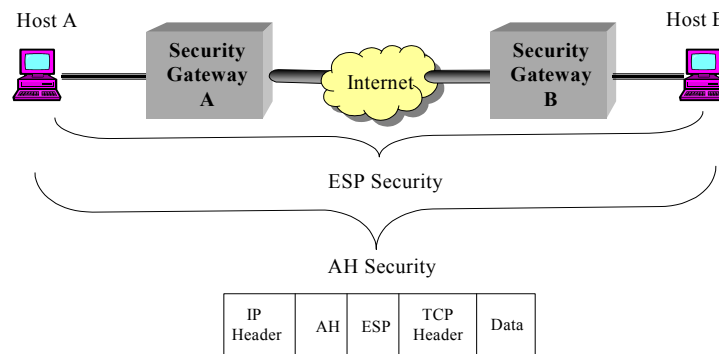
Figure 5.   Transport Adjacency.
(After: Leiseboer, John, 2001)

Here more than one security protocol is applied to the same IP packet, without invoking tunneling (Refer to Figure 5). This approach to combining AH and ESP allows for only one level of combination. The advantage of this approach over simply using an ESP SA with the ESP authentication option is that the authentication covers more fields, including the source and destination IP addresses. The disadvantage is the overhead of two SAs versus one SA.

b. *Iterated Tunneling:*

Multiple layers of security protocols are effected through IP tunneling (Refer to Figure 6). This approach allows multiple levels of nesting, since each tunnel can originate or terminate at a different IPsec site along the path. For e.g. having a transport

9

SA between two hosts could combine the SAs to travel part of the way through a tunnel SA between secure gateways.
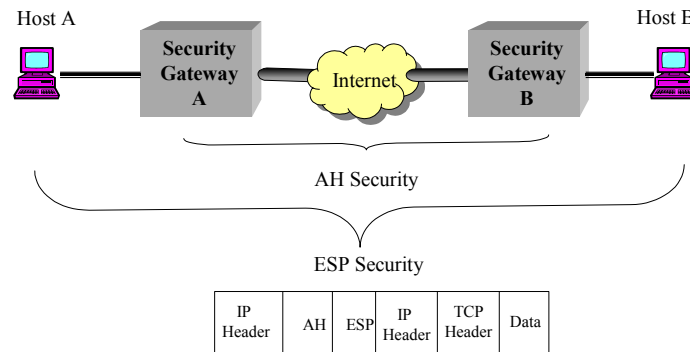
Figure 6.   Iterated Tunneling. (After: Leiseboer, John, 2001)

Four basic combinations of SAs are possible:

o   Security provided between end systems that implement IPsec.

o   Security is provided only between gateways (routers, firewalls, etc.) and no hosts implement IPsec.

o   Security provided both at gateways and hosts.

o   Security between remote host and a local gateway.

5.   Key Management

The key management portion of IPsec involves the determination and distribution of secret keys. A typical requirement is four keys for communication between two applications: transmit and receive pairs for both AH and ESP. The two types of key management are manual and automated.

The Internet Security Association and Key Management Protocol, (ISAKMP) provides a framework for automated Internet key management and provides the specific protocol support, including formats for negotiation of security attributes. ISAKMP by itself does not dictate a specific key exchange algorithm; rather ISAKMP consists of a set

of message types that enable the use of a variety of key exchange algorithms. IKE, adapted from Oakley, is the specific key exchange algorithm mandated for use with the initial version of ISAKMP. ISAKMP defines procedures and packet formats to establish, negotiate, modify and delete security associations.

B.   KEYNOTE TRUST MANAGEMENT SYSTEM

OpenBSD IPsec incorporates the concept of trust and security policy management by implementing KeyNote. The research performed in this thesis utilizes the OpenBSD IPsec mechanism as a model for discussion and implementation. Figure 7 depicts the KeyNote trust management process.

Trust management, is a unified approach to specifying and interpreting security policies, assertions[2], credentials[3], and relationships; it allows direct authorization of security-critical actions. A trust-management system provides standard, general-purpose mechanisms for both local and remote specification of application security policies and credentials. Trust-management credentials describe a specific delegation of trust and subsume the role of public key certificates; unlike traditional certificates, which bind keys to names, credentials can bind keys directly to the authorization to perform specific tasks. (From Blaze, Matt, Feigenbaum, Joan, and Keromytis, Angelos D., RFC 2704)

A trust-management system has five basic components:

o   A language for describing `actions', which are operations with security consequences that are to be controlled by the system.

o   A mechanism for identifying `principals', which are entities that can be authorized to perform actions.

o   A language for specifying application `policies', which govern the actions that principals are authorized to perform.

---

[2]   An assertion is a statement binding together an authorizing principal, an authorized principal(s) and a set of conditions.

[3] Credentials are similar to assertions, but each credential must be signed by the authorizer. Thus the authorizing principal of a credential must be a public key.

o A language for specifying `credentials', which allow principals to delegate authorization to other principals.

o A `compliance checker', which provides a service to applications for determining how an action requested by principals should be handled, given a policy and a set of credentials.

The trust-management approach has a number of advantages over other mechanisms for specifying and controlling authorization, especially when security policy is distributed over a network or is otherwise decentralized.

Trust management unifies the notions of security policy, credentials, access control, and authorization. An application that uses a trust-management system can simply ask the compliance checker whether a requested action should be allowed. Furthermore, policies and credentials are written in standard languages that are shared by all trust-managed applications; the security configuration mechanism for one application carries exactly the same syntactic and semantic structure as that of another, even when the semantics of the applications themselves are quite different.

Trust-management policies are easy to distribute across networks, helping to avoid the need for application-specific distributed policy configuration mechanisms, access control lists, and certificate parsers and interpreters.

KeyNote is a simple and flexible trust-management system designed to work well for a variety of large- and small- scale Internet-based applications. It provides a single, unified language for both local policies and credentials. KeyNote policies and credentials, called `assertions'; contain predicates that describe the trusted actions permitted by the holders of specific public keys. KeyNote assertions are essentially small, highly structured programs. A signed assertion, which can be sent over an un-trusted network, is also called a `credential assertion'. Credential assertions, which also serve the role of certificates, have the same syntax as policy assertions but are also signed by the principal delegating the trust.

In KeyNote:

o Actions are specified as a collection of name-value pairs. For instance a name value pair could be app_dom = "email". These are called as action attributes and a query is made with the action attributes and their associated values.

o Principal names can be any convenient string and can directly represent cryptographic public keys.

o The same language is used for both policies and credentials.

o The policy and credential language is concise, highly expressive, human readable, and compatible with a variety of storage and transmission media, including electronic mail.

o The compliance checker returns an application-configured `policy compliance value' that describes how a request should be handled by the application. Policy compliance values are always derived from policy and credentials, facilitating analysis of KeyNote-based systems.

o Compliance checking is efficient enough for high-performance and real-time applications.

Despite these advantages, the KeyNote Policy language has some technical challenges regarding understandability of complex policies, which are described in later sections.
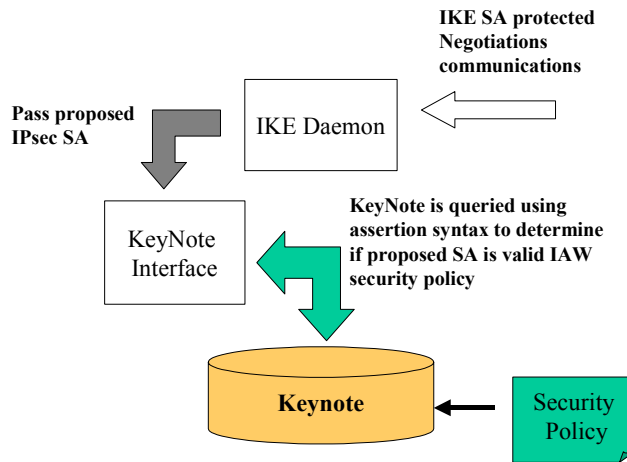
Figure 7.   KeyNote Process (After Agar, Chris December 2001)

C.   QUALITY OF SECURITY SERVICE (QOSS)

IPsec provides a high degree of granularity in discriminating between traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec. Further use of a trust management system such as Keynote enables an application to simply ask the compliance checker whether a requested action should be allowed. Thus if we specify a granular security policy as permissible by IPsec and use Keynote to verify a request based on the policy, we would be able to modulate the security settings of applications dynamically in accordance with the security and performance requirements of the applications in particular, and networks as a whole. This is the essence of 'Quality of Security Service' (QoSS).

In the Quality of Service (QoS) model, resource allocation is adjusted to meet user requests under changing network environment and resource availability conditions. Similarly, QoSS, see Figure 8, provides a mechanism to manage security services to meet requirements set by the user's requests, system's security policy, availability of system resources and network environment. (Irvine and Levin, September 2000)

Similar to the modulation of resources to support QoS, security services can be defined in terms of user and system requirements, network environment factors and available resources. Without a range of security services, a user is faced with the rigid

14

and limited choice of "all or nothing" for each security service. Historically, security services have been provided in such a static manner. (Spryropoulou, Agar, Levin and Irvine January 2002) Quality of Security Service (QoSS) provides a more flexible solution to the provision of security services. The security resource manager and/or the security system can adjust security service to meet user requirements, system security policy and network environment constraints. (Irvine and Levin, September 2000)

Security systems and managers can maintain overall control of the security mechanism through QoSS "system security policies." These policies dominate the individual "user security requirements." Specifically, they define all authorized operations per user, system, application, etc.

QoSS has several mechanisms to handle security variablity. A security variance for a particular policy exists when that policy may be enforced utilizing a specific range of security attributes. Therefore, based on the policy parameters, the attributes used to enforce the security policy may differ according to selection criteria such as "network mode". Fixed requirements are used to set minimum level acceptable security standards. A range of security settings meeting or exceeding this minimum level can be provided. For example a system may utilize SHA as a minimum level authentication algorithm for all message handling. Users or applications could apply further granularity in support of confidentiality to messages by selecting an encryption algorithm from a provided range. Other examples of variable security attributes that may be used are: assurance level, key length or security attribute expiration date stamp. (Irvine and Levin   September 2000)

Quality of Security Service (QoSS)

Security Resource Manager

Secure Communication
Negotiation

Secure Communication
Negotiation

User A   Negotiated range of secure communication attributes   User B
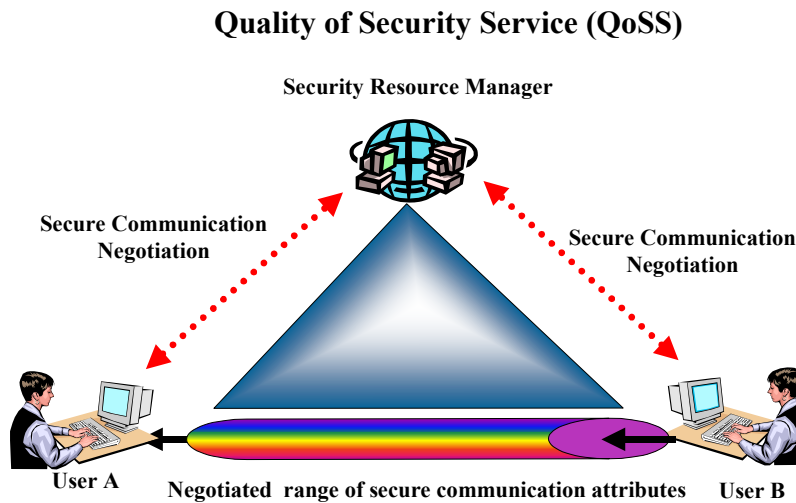
Figure 8.   Quality of Security Service (QoSS). (From Agar, Chris December 2001)

1.   Managing Quality of Security Service (QoSS)

Inevitably, security mechanisms result in a cost to the user, system and resources. Whether in the form bandwidth, algorithm processing time, overhead, or funds, the cost of security is a challenging concern to resource managers. A "costing framework" may be used to map security service resource consumption to available resources; ultimately enabling a management system to efficiently and effectively handle security service costs.

Security services, as previously described, may utilize high level services and consume lower level resources in a system. High-level services include, for example, non-repudiation, auditing, authentication, encryption, or intrusion detection. Low-level resources include memory, bandwidth, or processor time. Further, each security service will require a governing policy, consisting of specific rules that determine how and when to use the service. Therefore each network task associated with QoSS can be mapped to a vector of security requirements directly associated with the security services the task requires. (NPS-CS-02-001, January 2002)

a.   *Dynamic Parameters*

Government and DOD organizations utilize a variety of dynamic parameters to define a predefined response of specific actions according to policy.

16

Examples include INFOCON and THREATCON levels. In order for a security mechanism to be fully functional within the DOD and Government infrastructure, it has to be able to incorporate the dynamic parameters into the security setting decision-making process. A change in an INFOCON or THREATCON level should have an immediate effect on attributes and settings in a security mechanism. By introducing a dynamic mechanism, a system can modulate its security settings in response to these dynamic parameters. Security level and network mode, defined in the following sections, have been chosen as two abstract dynamic parameters that govern changes to security attributes as defined in the organization's security policy. (NPS-CS-02-001, January 2002)

By developing and implementing a security mechanism that can dynamically adjust in accordance with a change to network modes and/or security levels, the users and managers do not have to be concerned with the fine granularity and low-level complexity involved in adjusting and selecting appropriate security attributes, such as keylengths and cryptographic algorithms.

b.   User Choices for Security Levels

Security classification levels are a common metric used in the government and DOD to distinguish authorization for classified information. Common levels include Top Secret, Secret, Classified and Unclassified. Each of these levels correspond to different governing policies and requirements associated with the threat to national security by the disclosure of information to adversaries. Likewise, security selection levels, as defined here for proof of concept, represent an increasing requirement for stronger security (e.g. encryption and authentication algorithms).

Network security policies may utilize a range of maximum and minimum-security levels for each variant security service. Minimum-security levels set the lowest acceptable security attributes and maximum-security levels establish a ceiling on the use of available security resources. Intersections of policies require further granularity in security settings to satisfy all governing users and systems. A user may also desire to select a higher level of security than the predefined minimum. (NPS-CS-02-001, January 2002)

A user or application, however, may quickly become overwhelmed with the security setting details, potentially resulting in degraded security or performance. By developing security definitions that encompass detailed security settings required by users or applications, the complexity of the selection process for the security settings can be simplified to a reasonable level. One approach would involve the use of the following Network Security levels: high, medium and low. (NPS-CS-02-001, January 2002) 'High' security level would utilize strong levels of security attributes, medium level, moderate level of security attributes, and low level, low to no security attributes.

By implementing this approach the system security resource manager or security engineer is responsible for presetting security variables and ranges in accordance with choices offered to users or applications. A mapping of allowable security settings to security levels, providing a range of selection or specific values, is required to properly enforce the system security policy. (NPS-CS-02-001, January 2002)

c. The Notion of Network Modes

Networks exist in a variety of states, providing users and systems with varying levels of service. On one occasion the network may experience heavy levels of traffic resulting in a poor performance. At other times the network may be limited in the availability of resources due to maintenance, and at other times the network may be performing at its optimum level. To fully incorporate the performance and reliability of the network into a Quality of Security Service mechanism, the notion of network modes is introduced.

There are numerous situations in which a network security policy will be required to dynamically change to properly address the current operational threats and needs, as well as the availability of resources and network performance. In the midst of a highly sensitive intelligence operation transmitted reports will require the highest possible security to ensure the information and the sources remain protected. In another scenario, an enterprise confronted with a serious emergency that requires the fastest possible transmissions may not be concerned with transmission protection. (NPS-CS-02-001, January 2002) Therefore a requirement exists for a dynamic security mechanism that can appropriately adjust to meet the needs of the system, users or applications. One

approach is to use the following network modes: normal, impacted, and crisis. Normal mode is defined as ordinary operating conditions with normal traffic load and no heightened threat conditions. Impacted mode may be defined when the network/system is experiencing high levels of traffic and therefore certain security selection may not be available due to efficiency constraints. Emergency mode may be defined as a situation that requires the highest level of security or the lowest level dependent on the situation and policy. (NPS-CS-02-001, January 2002)

d. *Mapping Abstract Parameters to Security Mechanism*

A mapping of abstract dynamic parameters to resident security mechanisms is required to properly enforce policy decisions. For example, network modes may be mapped to security level ranges and ultimately to security attributes and settings.

Network Mode	Security Level	Security Attributes
Normal	Low	ENCRYPTION: NONE AUTHENTICATION: MD5
	Medium	ENCRYPTION: DES AUTHENTICATION: MD5
	High	ENCRYPTION: 3DES AUTHENTICATION: MD5
Crisis	Low	ENCRYPTION: NONE AUTHENTICATION: NONE
	Medium	ENCRYPTION: NONE AUTHENTICATION: NONE
	High	ENCRYPTION: DES AUTHENTICATION: MD5
Impacted	Low	ENCRYPTION: 3DES AUTHENTICATION: MD5
	Medium	ENCRYPTION:3 DES AUTHENTICATION: SHA
	High	ENCRYPTION: AES AUTHENTICATION: SHA

Broad	Granularity	Fine
Users		System Admin/ Security Experts

Figure 9. Mapping Security Policies to Security Attributes.
(From Agar, Chris December 2001)

The security resource manager and security engineer would define the network modes and security levels to provide the users and applications with appropriate

19

security service as translated into QoSS choices. Once defined, the complexity of the security mechanism and security attribute selection is transparent to the user. (See Figure 9)

2. Implementation Issues

Quality of Security Service (QoSS) provides us with a mechanism to modulate the security settings and enhance performance based on both necessity (e.g. threat) and resource availability. It also provides us with a tool to ensure that the minimum-security requirements of applications and the network as proposed in the security policy is not violated. Hence defining an adaptive security policy based on network threat and performance conditions is the key to optimal and secure utilization of the network resources. Keynote provides one such policy specification language but the complexity of the language makes its practical implementation extremely difficult. An abstraction for this language is therefore felt necessary. It would use the power of Keynote for formal compliance checking and at the same time be easy to use and administer. This is dealt with in the following chapters in detail.

III. XML AND KEYNOTE POLICY LANGUAGE

A. KEYNOTE POLICY SPECIFICATION LANGUAGE

1. Introduction

The syntax and semantics of the Keynote language is described in detail in RFC 2704. (Blaze Matt, Feigenbaum, Joan, Ioannidis, John, and Keromytis, Angelos D). In this section a brief overview of the language and the specific parts that need emphasis will be highlighted. The language is used for specifying application 'policies,' which govern the actions that principals (entities that can be authorized to perform actions) are authorized to perform. The language provides the semantics for describing 'actions,' which are operations with security consequences that are to be controlled by the system. It is also used for specifying 'credentials', which allow principals to delegate authorization to other principals.

2. Keynote Assertion Syntax

a. Basic Structure

Keynote assertions are divided into sections, called 'fields' that serve various semantic functions. Each field starts with an identifying label at the beginning of a line, followed by the ":" character and the fields contents. There can be at most one field per line.

A field may be continued over more than one line by indenting subsequent lines with at least one ASCII SPACE or TAB character. Whitespace (a SPACE, TAB, or NEWLINE character) separates tokens but is otherwise ignored outside of indentation and quoted strings.

One mandatory field is required in all assertions:

- Authorizer

Six optional fields may also appear:

- Comment

- Conditions

- KeyNote-Version

- Licensees

- Local-Constants

- Signature

All field names are case-insensitive. The "KeyNote-Version" field, if present, appears first. The "Signature" field, if present, appears last. Otherwise, fields may appear in any order. Each field may appear at most once in any assertion. Blank lines are not permitted in assertions. Multiple assertions are stored in a file (e.g., in application policy configurations), therefore, they can be separated from one another unambiguously by the use of blank lines between them.

For the most part it is the conditions field that has many variables and a typical policy file will have a detailed conditions section. We shall examine the conditions field in detail as it applies to our application.

b. Conditions Field

The field gives the 'conditions' under which the Authorizer[4] trusts the Licensees[5] to perform an action. The exact semantics of the field is defined in RFC 2704. However parts of the language pertinent to our application are explained below.

Security attributes reside in the conditions section and are expressed in the form of logical statements. The conditions section's syntax is similar to that of a programming language "if statement". The section is usually broken into sub statements by using &&, ||, and parenthesis to construct logical conditions. For example the following phrase describes two security proposals supporting Telnet services (service_port= 23) using ESP with 3DES for encryption and finger services (service_port=79) using AH with SHA for authentication:

(local_filter_port == "23" &&

---

[4] The Authorizer identifies the Principal issuing the assertion.

[5] The Licensees identifies the principals authorized by the assertion. More than one principal can be authorized, and authorization can be distributed across several principals through the use of `and' and threshold constructs.

esp_present == "yes" &&

esp_enc_alg == "3des") ||

(local_filter_port == "79" &&

ah_present == "yes" &&

ah_auth_alg == "sha") -> "true";

3.   Keynote Policy File

A simple policy file may contain very few elements. For instance we consider an hypothetical application called "SPEND" wherein the authority is delegated authority to RSA key dab212. Further a condition is specified to permit delegation only when amount given in the "dollars" attribute is less than 10000. The policy would then look as under:

```
Authorizer: "POLICY"
Licensees: "RSA:dab212"  # the CFO's key
Conditions:(app_domain=="SPEND")&&(@dollars
< 10000);
```

A more detailed policy is listed in Appendix 'D'.

4.   Keynote Policy File with Quality of Service Parameters.

Using the example in section 2(b) above, with security levels "high" and "low" and network modes "normal" and "impacted", the condition phrase is expanded. (From Agar, December 2001)

KeyNote-Version: 2

Comment: Policy file for Network Modes and Security Levels

Authorizer: "POLICY"

Licensees: "passphrase:mekmitasdigoat"

Conditions: ((app_domain == "IPsec policy") && (

((network_mode = "normal" &&

((security_level = "high" &&

((local_filter_port == "23" &&

esp_present == "yes" &&

esp_enc_alg == "3des") ||

(local_filter_port == "79" &&

```
                    ah_present == "yes" &&

                    ah_auth_alg == "sha"))) ||

              ((security_level = "low" &&

               ((local_filter_port == "23" &&

                    esp_present == "yes" &&

                    esp_enc_alg == "des") ||

               (local_filter_port == "79" &&

                    ah_present == "yes" &&

                    ah_auth_alg == "des-mac")))) ||

         (network_mode = "impacted" &&

              ((security_level = "high" &&

               ((local_filter_port == "23" &&

                    esp_present == "yes" &&

                    esp_enc_alg == "aes") ||

               (local_filter_port == "79" &&

                    ah_present == "yes" &&

                    ah_auth_alg == "sha"))) ||

              ((security_level = "low" &&

               ((local_filter_port == "23" &&

                    esp_present == "yes" &&

                    esp_enc_alg == "3des") ||

               (local_filter_port == "79" &&

                    ah_present == "yes" &&

                    ah_auth_alg == "sha-md5")))) -> "true";
```

As we notice the complexity of the language increases exponentially as we add more ports and parameters to it. The nesting of parenthesis to multiple levels makes writing a syntactically correct policy file almost impossible. In the following section, XML is analyzed to see if the technology could be used to make the task of specifying the Keynote policy file practical.

B. XML

1. Introduction

Extensible Markup Language (XML) is a rapidly maturing technology with powerful real-world applications, particularly for the management, display and organization of data. XML is a technology concerned with the description and structuring of data. It is a subset of Standard Generalized Markup Language (SGML), with the same goals, but with much less complexity. XML is not a language but a standard for creating languages that meet the XML criteria. It describes a syntax that you use to create your own languages (David Hunter, Kurt Cagle, Chris Dix, Roger Kovack, Jonathan Pinnock, Jeff Rafter February 2002).

Data is separated from presentation in XML. XML structures the data, while style sheets format the data presentation. That makes it easier to use the data for multiple purposes. The same stylesheet can be used with multiple documents to create a similar appearance among them. Or alternatively multiple stylesheets can be applied to an XML document to provide different forms of presentation of the data. There are a variety of languages that can be used to create stylesheets such as Extensible Stylesheet Language Transformations (XSLT).

XML solves the problem that has been the focus of attention for many years now, data portability and software maintenance. Programmers have been structuring their data in an infinite variety of ways, and with every new way of structuring data comes much experimentation and testing to get it just right. If the data format changes, the methodologies to manipulate it also have to change, and the testing and tweaking has to begin again. The cycle of software maintenance will start all over again. With XML, there is a standardized way to structure the data and to extract the information we need. The extensibility of the language permits us to make changes, as we need without having to tweak adjust the code that extracts information from the file.

2. XML Parsers

XML documents have to be 'well-formed'. Well-formed XML is XML that meets certain syntactical rules outlined in the XML 1.0 specification. Describing the documents

following these rules enables general-purpose XML programs called XML parsers to access the data from the document. XML parsers are programs that are able to read XML syntax and extract the tag names and the values i.e. the attributes and their values. There are two types of parsers, validating and non-validating parsers. Validating parsers in addition to parsing the data, also validate the data against the specified XML Schema or DTD. This ensures that the structure of the XML document as well as the data within it adheres to a predefined agreement or specification.

There are a wide variety of well tested, industry strength parsers that we can use within our applications to access XML data. Thus applications designed to handle XML can inherit the properties of a well-tested parser and would not need to write all the code from scratch. Thus changes in the data format, as long as the data follows the XML standards, will not affect the application code. The parsers handle that for us. Even the language in which XML is written does not matter to the parsers. Thus if we create an XML document we can be sure that any XML parser will be able to retrieve information from that document, even if we can't guarantee that any application will be able to understand what that information means. Examples of parsers include, Microsoft XML Parser (MSXML), Apache Xerces parser, IBM's Xml4j parser, DOM, SAX etc.

3. XML DTDs

The need to validate documents against a vocabulary led the creators of XML to include a method of checking validity in the XML recommendation. A document is valid if its XML content complies with a definition of allowable elements, attributes and other document pieces. By utilizing special 'Document Type Definition' syntaxes or DTDs, you can check the content of a document type with a special parser. The Document Type Definition (DTD) validation format has been used for many years to validate SGML and XML documents. The XML recommendation separates parsers into two categories – validating and non-validating. Validating Parsers, according to the recommendation, must implement validity checking using DTDs., Therefore if we have a validating parser, we can remove the content checking code from our application and depend on the external processor. For instance you may use an XML software tool such as 'XML-Spy' to perform this check for you. Thus by utilizing DTDs, we can easily validate our XML documents against a defined vocabulary of elements and attributes. As the use of XML

and DTDs increased, some of the limitations of DTDs surfaced. Though these limitations restrict their use, DTDs are still useful for various applications. Some of the limitations of DTDs are:

- o DTDs were developed long before XML became a popular data transfer format. As a result, they do not follow the XML rules as XML schemas (to be described later) do.

- o Support for XML Namespaces: Namespaces are used frequently in different types of XML documents to prevent naming conflicts. This allows elements that are used in different contexts to be combined without mixing up the meaning of the elements. By definition, an 'XML namespace' is a collection of names, identified by a Uniform Resource Identifier (URI) reference [RFC2396], which are used in XML documents as element types and attribute names. DTDs have no concept of namespaces. In fact, most validating XML parsers allow you to "turn off" namespaces while validating an XML document against a DTD. XML schemas, on the other hand, fully support namespaces and do so well.

- o Poor data typing: DTDs have no real concept of data types. In fact, when you define an element in a DTD that contains a text node, you can only specify that the text node is Parsed Character Data (PCDATA). You cannot specify that the text must be a decimal, integer, date, and so on. Although DTD attribute definitions do contain a few more built-in data types such as ID, IDREF, and NMTOKEN, they still do not allow for validating against data types found in many relational databases. In contrast, XML schemas provide robust support for data types and also allow data types to be extended and customized.

- o Lack of the property of inheritance.

However, with the release of XML schemas, a more powerful mechanism for validating XML documents is now available.

27

4.   XML Schemas move to next page

A Schema is the XML construct used to represent the data elements, attributes, and their relationships as defined in the data model. By definition, a DTD and a schema are very similar. However, DTDs usually define simple, abstract text relationships, while schemas define more complex and concrete data and application relationships. A DTD doesn't use a hierarchical formation, while a schema uses a hierarchical structure to indicate relationships. XML Schema definitions are also commonly referred to as XSD. Some of the benefits of XML Schemas are:

o   XML Schemas are created using XML, not an alternative SGML syntax.

o   XML Schemas fully support the Namespace Recommendation. The goal of the W3C XML namespaces recommendation was to create a mechanism in which elements and attributes within an XML document that were from different markup vocabularies could be unambiguously identified and combined without processing problems ensuing. The XML namespaces recommendation provided a method for partitioning various items within an XML document based on processing requirements without placing undue restrictions on how these items should be named. (Namespaces in XML, Jan 1999).

o   XML Schemas allow you to validate text element content based on built-in and user-defined data types.

o   XML Schemas allow you to more easily create complex and reusable content models.

o   W3C XML Schema borrows a number of concepts from object oriented programming including the notions of abstract types, type substitutions, and polymorphism. Abstract elements and substitution groups allow schema authors to create or utilize schemas which define generic base types and extend these types without affecting the original schema.

Data types in an XML Schema definition are of two broad categories: simple and complex. Elements that may contain attributes or other elements are 'complexTypes',

28

Attribute values and elements that contain only text content are 'simpleTypes'. More commonly used simpleTypes, such as 'int', 'float' etc are built into XML Schemas, Data typing thus provides a rigid control on the input data. An XML document that adheres to a particular XML Schema is an XML Schema 'instance' document. Validating a document against an XML Schema requires the use of a special parser (Refer Figure 10). The XML Schema Recommendation calls these parsers 'schema validators'.
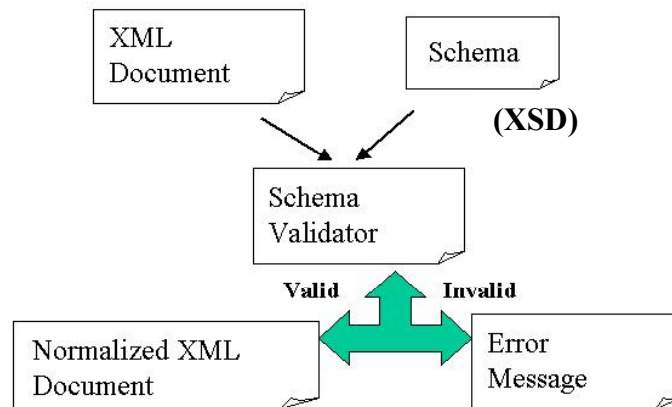
Figure 10. Document validation with Schemas.

5. XSLT

XML lets us structure our data in a hierarchical structure. This structure has some rigid rules and following them enables us to use other XML tools to access and manipulate the data without having to write code for it. However the structure may not suit an application and we may need an alternative representation of the data for either presentation purposes or for the purpose of manipulating it. Extensible Stylesheet Language Transformations, XSLT, is a language which can transform XML documents into any text-based format, XML or otherwise. It is a sub-component of a larger language called XSL. XSL relies on finding parts of an XML document that match a series of predefined templates, and then applying transformation and formatting rules to each matched part. Thus once an XML document is created, XSLT can be used to transform

29

the document into whatever other format we wish- HTML for display on web sites, a different XML-based structure for other applications, or even just regular text files.

XSL is used to create stylesheets. An XSL engine uses these stylesheets to transform XML documents into other document types, and to format the output. Stylesheets define the layout of the output document and the location of the data in the source document. That is, "retrieve data from this place in the input document; make it look like this in the output". In XSL parlance, the input document is called the source tree, and the output document the result tree.

XSLT is a declarative programming language as opposed to imperative programming languages like C++ or Java. It has no side effects and it has mechanisms to understand XML and HTML formats. XPATH is another W3C language specification that is used in conjunction with XSLT that allows us to address specific parts of an XML document and get the specific pieces of information that we need.

6. Advantages of XML for the Policy Specification Language

As described in Section IV we have a need to represent the intended IPsec policy in a form separate from the native KeyNote representation. Some of the advantages that would accrue by using XML are as follows:

a. Tools

Use of XML for specification of the KeyNote policy file lends itself to be used with the freely available, verified, tested and user-friendly tools. These tools include among others, XML editors, parsers, validators, translators etc. The availability of such tools and the extensive use of XML in modern communication protocols and other programs will enable users to manipulate XML files easily. Wide availability of such tools will also help in creating and maintaining the policy files over diverse systems without the need for an application specific editor.

b. Security

Interest in XML in recent years has resulted in huge investments in the field of XML security. The XML security features such as XML encryption and authentication will enhance the security of the policy file. This will also help, for

instance, in selectively 'digitally signing' parts of the policy file. Thus a person signing a particular part of the policy file will only be responsible for the part he signed. Without the XML format it would be possible only to sign the entire file after for instance adding a part to it.

c.   Platform Independence

It is possible to edit, maintain and distribute the XML policy file across different OS platforms.

d.   Single Data Multiple Presentation

 Once we represent the policy in an XML format it is possible to extract relevant information and present it in different forms that are more intuitive and useful to the administrator or the user. XSLT style sheets can be written and associated with the policy file to generate different presentation formats. Apart from presenting it in a more understandable and probably graphic format this will also help the administrator pin down any inaccuracies/inconsistencies/contradictions in the policy file. Intelligent agents can be written to audit the policy file and signal the administrator for errors in the policy file.

e.   Consistency and Accuracy

XML Schemas and/or DTDs can be used to validate the XML file to see if it matches our specifications. Validating the policy file with a well-defined schema will enable errors to be picked up. This will trap all errors without having to go through the entire file manually. The use of generic schema generators and validators only makes this an easier task. This will also enable users to verify policy files received across the networks.

f.   Extensible Format

An XML format will lend itself to extend the policy file to cater to new requirements in the policy file that come up in the future. Additional tags can be defined for elements and attributes as and when the need to incorporate them arises. This would not require changes to the application code as long as the structure of the document is maintained.

g.   Ease of Use Move down

The hierarchical nature of XML layout results in an easy to use and easy to manipulate format. It makes the file more modular and so easily understandable.

h   *Semantic Content Use:*

The semantic content of the policy file enables future deployment of intelligent agents or roaming agents that can read policy files and report problems, and that can resolve conflicts between multiple systems by highlighting for instance the difference in the policies between them.

C.   INTEGRATING XML AND KEYNOTE POLICY

The Keynote engine requires that the assertions, credentials and the policy files be specified in the syntax as specified in RFC 2704 and examined in section 'A' above. This structure restricts our ability to define any meaningful network security policy in an error free manner. Further, any policy file received in this format is not human readable, thus establishing a daunting requirement to verify its correctness and to detect security loopholes if any. Thus there is a clear problem of differentiation between data content and its representation. The same data is required by the Keynote engine in one format while on the other hand the format is not suited for human interpretation and validation. This is where it is felt that XML could be brought in. Specifying the policy data in an XML format will enable us to use XSL to translate the data to the format needed. XSL could also be used to present the data a more human readable form. Further specifying an XML Schema would provide us the benefit of validating the XML policy file for correctness prior to its transformation.

IV. DESIGN AND IMPLEMENTATION

A. DESIGN METHODOLOGY

The design of the policy editor was done in the following major steps:

1. Study the existing keynote policy language.

2. List out the alternative approaches to design.

3. Examine XML and evaluate design alternatives using XML.

4. Design of a Java based GUI.

5. Integration of various components.

B. ALTERNATIVE DESIGN APPROACHES

Two design approaches were studied and implemented. These are as below:

1. Option 1: Non XML Version

We shall look at the 'conditions' field of the policy file in the following discussion. The first option considered was to use any high level programming language to develop a graphical user interface that takes user inputs and converts it into the Keynote policy. This approach is extremely difficult to implement due to the complexity of the policy specification language. However preliminary research led to the observation that representing the user's policy data in a consistent data structure would enable easy processing of the data for writing out the Boolean expressions of the KeyNote file. The appropriate data structure for one such implementation and the algorithm for the application code is given in the succeeding paragraphs.

a. Data Structure

A 'tree' data structure is selected for the storage of the choices for the conditions field of the policy file (Refer Figure 11). A constraint wherein the children of any node meet an OR condition is enforced. The parent is then 'ANDed' to the 'ORed' children. For example, a condition specified as below would have an equivalent tree structure as shown:

o Condition Statement

((security_level=='high')&&((remote_filter_port ==21)|| (local_filter_port == 21)) ..
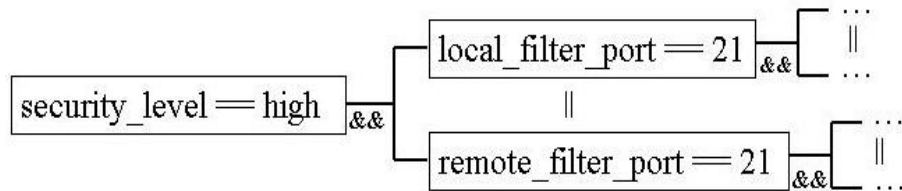
o Tree structure

Figure 11.   Tree Layout of a Condition Statement.

b.   *The Algorithm:*

An algorithm for generating KeyNote policy file from the user's tree structured policy data was implemented in Java. The algorithm is as below:

```
Function Generate_policy: Parameters Node, Returns String
// this is a recursive program that generates a document in the KeyNote
// format reading the policy data from a tree structured policy file.
Begin
        If node = = leaf node
        Begin
                Read attribute name
                Read attribute value
                Set node_string = (attribute name == ' attribute value' )
                Return node_string
        End
        Else
         Begin
                Read attribute name
```

34

Read attribute value

Set parent_string = (attribute name == ' attribute value')

Set child_count =0;

For each child_node

    Begin

        child_count = child_count+1;

        If child_count ==1 //first child

            first_child_string
                = Generate_policy(child_node);

        ElseIf child_count ==2

        Begin

            child_string
                = Generate_policy(child_node);

            Set   combined_string   =
            (first_child_string || child_string

        End

        Else

        Set combined_string = combined_string ||
        child_string

        EndIf

    End

    Set node_string = (parent_string && (child_string)

    Return node_string;

  End

  EndIf

End

c.   *Advantages of the Solution*

This program produces the desired result and has a few merits as under:

o   This solution results in simple application code. The code is mainly the recursive code along with other graphic components.

o   The complexity of the code is O(log n) , where n is the number of leaf nodes. This is due to the representation of the policy in a tree structure.

However since the tree need not be balanced it may not be extremely efficient. (Aho, Hopcroft and Ullman, 1987)

d. *Shortcomings of the Solution*

o The constraint imposed on the data structure works well with our current need of the policy file but may not suit all the future needs.

o It may be hard if not impossible to represent any arbitrary policy in a format that maintains the structure constraint.

o It does not solve the problem of auditing, consistency checks and validation of the stated policy. As the final form of the file is in keynote policy format, it is not possible to analyze the content of the file in a practical manner

e. *Implementation and Evaluation*

o The solution was implemented in Java and successfully generates the desired keynote policy file.

o Due to the major drawbacks of the solution of extensibility and validation the solution is not considered a viable practical solution though it meets the current needs.

2. Option 2: Defining an XML Policy File Format

The alternative was to look at an alternate representation of the policy logic in the form of a " user policy file". The research in coming to the algorithm for option one demonstrates the utility of a tree structured users policy file format. Further, the typical hierarchical nature of the policy file also points towards a tree structure. Use of XML thus seemed the logical step forward. Hence the overall approach for this option was to come up with format for the user policy file and then possibly transform that to the native KeyNote policy file format. This approach also provides the flexibility of extracting useful administrative information from the user policy file. All the advantages of using XML as described earlier in Section III.B.6 would also apply.

C.   XML POLICY FILE

Arriving at a format for the user policy file is a challenging task and there are multiple options available. The only fixed requirement is that the resulting XML file be well formed. During the course of the thesis research, multiple formats were designed. Each format had its strengths and shortcomings. For instance one format would lend itself to an easy application design while another would add more semantic content in the file format. The former therefore makes it easier to write an application such as a 'Policy Editor' while the latter results in a more descriptive self-defining file, which could be a good interchange format between multiple applications for instance. However after multiple iterations of modifications to the file format it is my opinion that the format does not matter as long as it has enough semantic content to make it understandable to the reader. I say this, as the choice of element tag names, their sequence etc. is a personal preference and the power of XSL is always available for another user who wishes to use an alternative format to transform the file to the format that he desires. Thus arriving at a well annotated, self-defining and logical policy file format was the endeavor.

Several choices had to be made. For instance a conditional statement like, (network_mode == 'normal' && (security_level =='high')) could be represented in many different formats. If the operators are implicit in the representation then we could for instance represent it in formats as in option 1-3 below or to reveal various types of Boolean operators we could follow the fourth or fifth option.

- o Option 1 : Use of PCData (Text content)

  <network_mode>

  normal

  <security_level>

  high

  </security_level>

  </network_mode>

o Option 2 – Use of Attributes

<network_mode value =’normal’>

    <security_level value = ‘high’>

    </security_level>

 </network_mode>

o Option 3 – Generic nodes

<element tagname = ‘network_mode’ value = ‘normal’>

    <element tagname = ‘security_level’ value = ‘high’>

    </element>

</element>

o Option 4 – Define the operators to be used. To write a policy file to represent a complex Boolean expression such as – ((network_mode == ‘normal’)&& ((security_level == ‘high’) || (security_level == ‘low’))), we could represent the same using XML as under:

<operator value = ‘&&’> <!--Note prefix usage -->

    <network network_mode = ‘normal’>

        <operator value = ‘||’>

            <security security_level ==’high’/>

            <security security_level ==’low’/>

        </operator>

    </network>

 </operator>

o Option 5 – Infix notation

<network network_mode = ‘normal’>

<operator value = ‘&&’>

```
<security security_level =='high'/>

                <operator value = '||'/>

<security security_level =='low'/>

</operator>

</network>
```

These are only a few possible representations of the file format. The options 1-3 make the application code simpler by making the operators implicit in the representation. However they do not lend themselves to extending to future needs by allowing other Boolean operations. The fourth and fifth operations make it possible to extend the format to any possible policy representation. However the writing the application code would require a little more detail and effort. As stated earlier any format is acceptable provided we support translations to the formats desired and that is the power of XML. Representation of the data should not hinder its use in any way.

The XML policy file that was finally arrived at is shown in Appendix 'A'.

D.   XSL

Having arrived at the XML policy file format XSL stylesheets had to be written to transform the policy file into desired formats. Two stylesheets were designed using XSLT (Refer Figure 12).
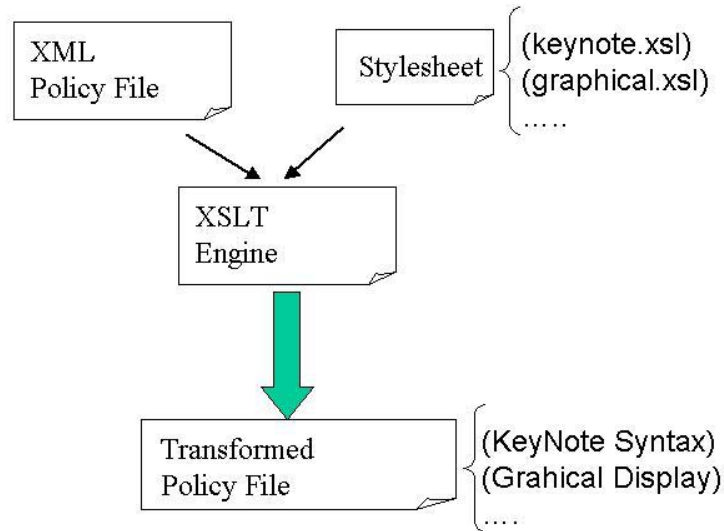
Figure 12.   XSL Transformation of XML Policy Data in the User Policy File.

The stylesheet for transforming the file to the Keynote policy file format is as shown in Appendix 'B'. The result of transforming the XML policy file this stylesheet is shown in Appendix D. An alternative stylesheet to transform the XML policy file to a more human readable, graphical web based format was also written. This is provided in Appendix C. The transformed output of the XML Policy file using this template is shown in Figure 17.

E.   JAVA BASED GUI

Though XML provides us with the flexibility to edit the policy file in any XML editor, it would still be convenient to provide a graphical user interface to manipulate the policy file. This would help in eliminating inadvertent errors and would also provide an automated entry into multiple elements without the need to edit each element content. This would enable global policy decisions to be applied throughout the policy file. An experienced system administrator could still capitalize on the use of the XML policy format and edit the file in the absence of the graphical user interface (GUI).

A Java-based GUI was therefore built to integrate various components of the software. Drop down menus and dialog boxes guide the user to input various parameters

required for the policy file. To enable maintenance of the GUI, from now on called the Policy-Editor, a separate XML configuration file was used to feed the data for various drop down menus and combo/list boxes. This decoupling of the Java code from the configuration data will enable continued use of the Policy-Editor without the need to modify the Java code.

The application uses the Java Document Object Model (JDOM) and Simple API for XML (SAX) packages to read, modify and translate XML files. JDOM beta 8 release was used and is available from http://www.JDOM.org

The application has been broadly divided into two modules the Admin module and the Operational Settings module. The Admin module caters for changes made infrequently and which provide the administrator to enter major changes to the policy file easily. The changes made in this menu option are saved in the config.xml file and will also result in changes to the entries in the combo/list boxes that appear under the operational settings. In other words, only entries that are allowed by the admin module will be available for further granular changes in the operational settings module. For instance if port 21 is opened using the Port-Management menu option under the admin menu then it would be possible to make further granular changes to Port 21 using the operational settings options. This may be for instance enable ESP with DES encryption for port 21.

Figures 13 to 16 are screen shots of the Policy Editor. Figure 13 and Figure 14 use the admin module to select ports and operational modes and security levels respectively. Figures 15 and 16 show the granular settings of encryption and authentication for particular ports. Figure 17 shows how the XSL transformation of the policy file displays the policy file in a graphical and more intuitive format.
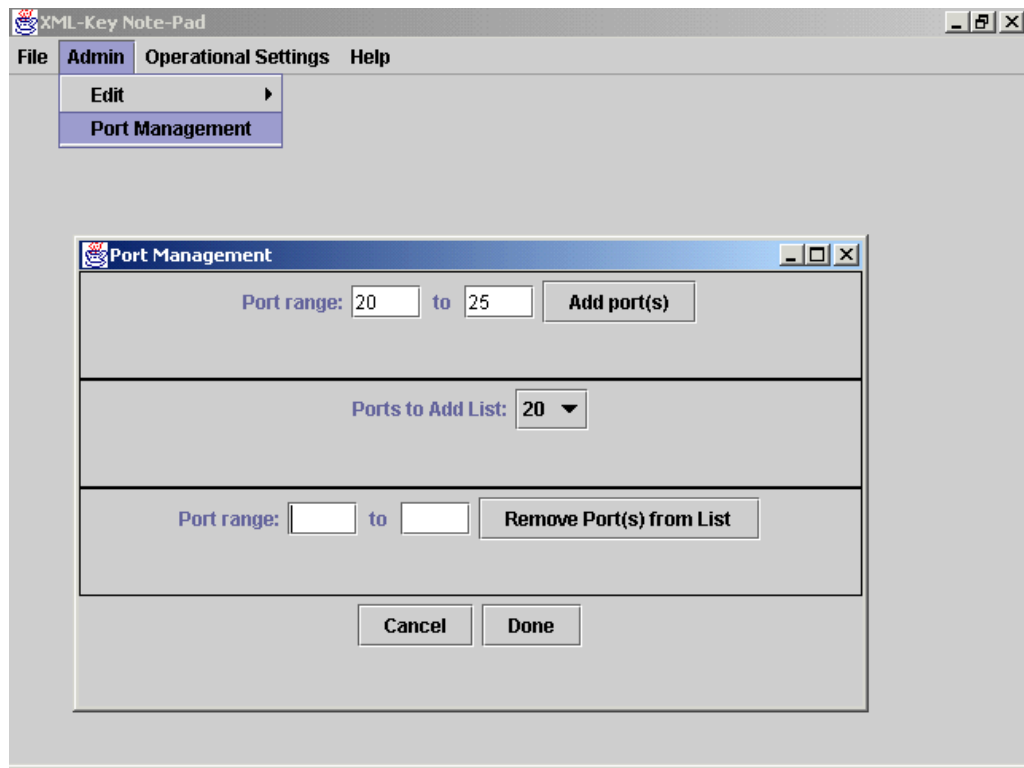
Figure 13.   Managing Ports in the Admin Module.
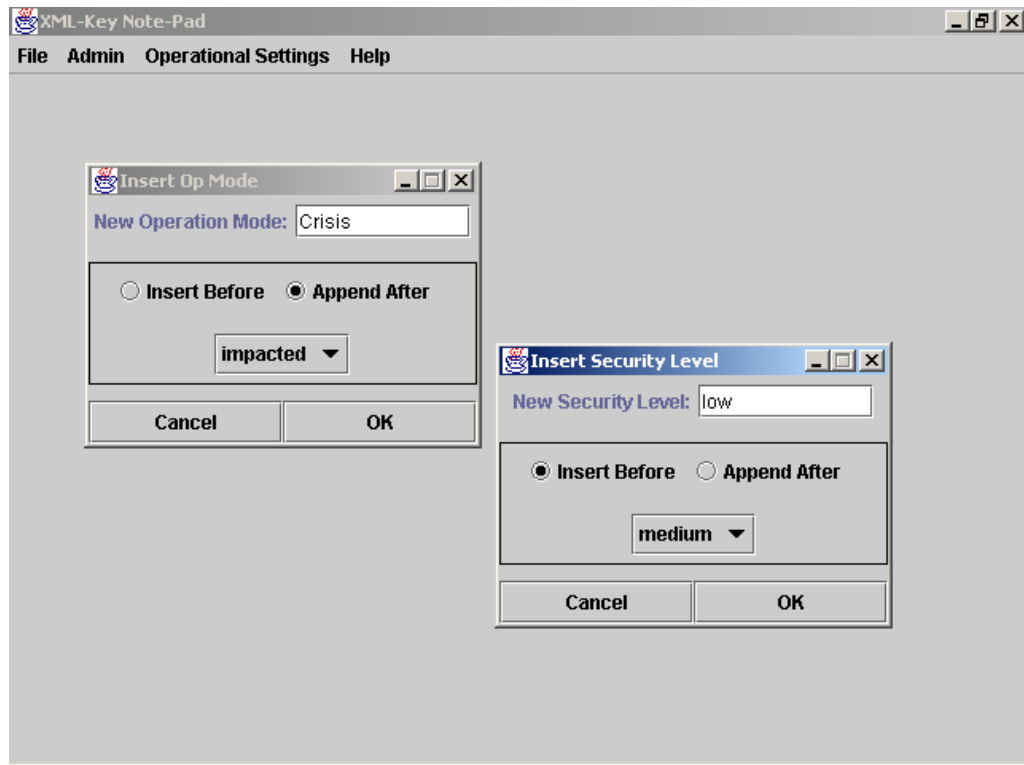
Figure 14.   Admin Mode Settings for Security Level and Op Modes
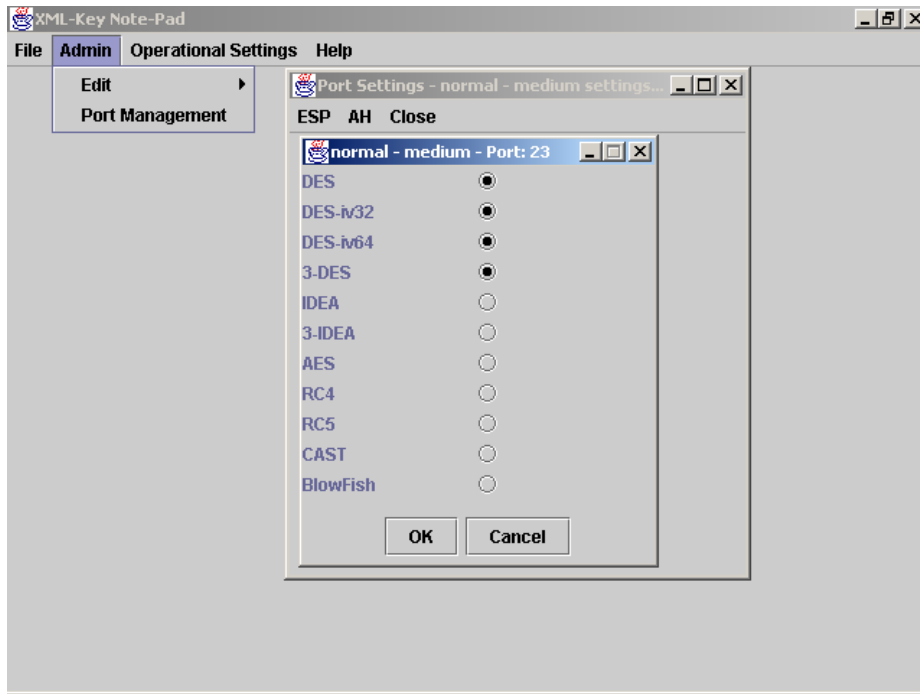
Figure 15.   Encryption Settings For Individual Ports
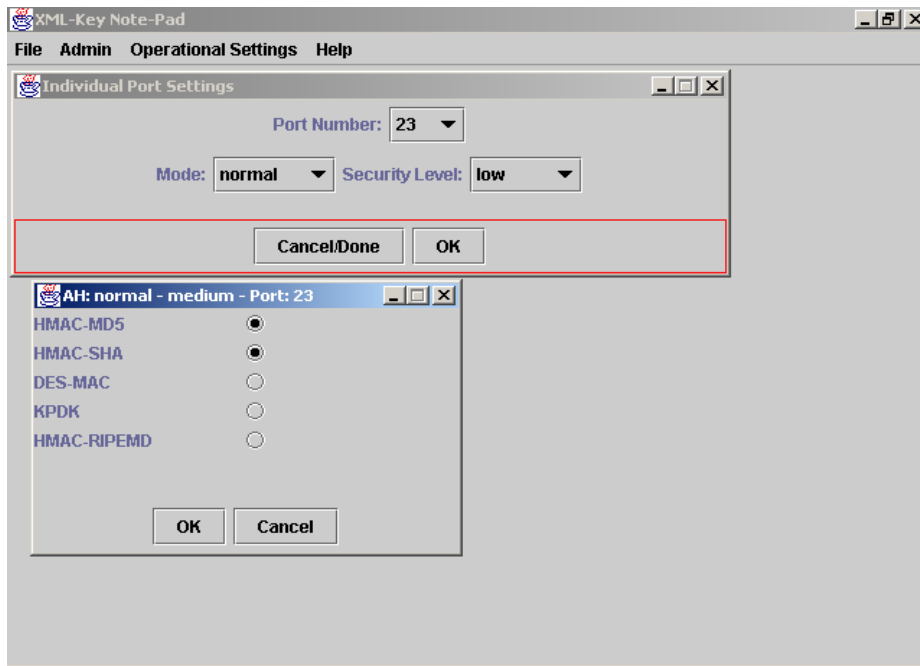
Figure 16.   Authentication Settings for AH Mode Heading should be with figure

F.   XML SPY – AN XML EDITOR

XML Spy Suite from Altova, Copyright ©1998-2002 Altova GmbH, is a comprehensive and easy-to-use product family that facilitates all aspects of XML Application Development. The product family consists of the XML Spy Document Framework and XML Spy IDE. XML Spy Document Framework consists of XSLT Designer and XML Spy Document Editor. The XSLT Designer enables writing of complex XSLT Stylesheets using an intuitive drag-and-drop user interface. XSLT Designer creates forms for use with XML Spy Document Editor. XSLT Designer was however not used for the purpose of writing the stylesheets for the policy editor, as they were too complex to be handled by the Designer. XML Spy Document Editor is a word processor type editor, supporting electronic form-based data input, graphical elements, tables, as well as real-time validation using XML Schema. XML Spy IDE is an integrated solution for XML-based application development, allowing easy creation and management of XML documents, XML schemas, as well as XSLT Stylesheets XML-Spy was used in the development of the XML Policy file. Figure 18 is a screen shot depicting the use of XML Spy editor. It displays how in the absence of the policy editor, the XML Spy (or any such editor) could be used to manipulate the XML Policy file directly. The result of schema validation can also be seen here. Figure 19 is the design view of the schema when viewed in XML Spy.
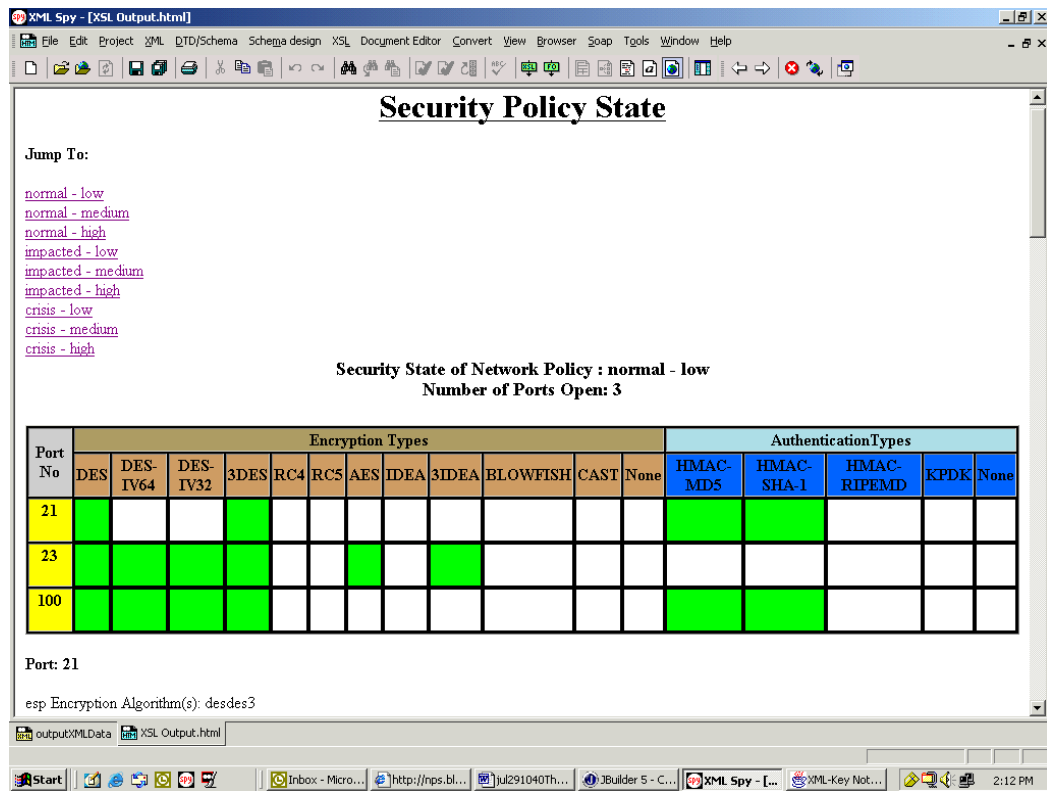
Figure 17.   XSL Transformation of the Policy File
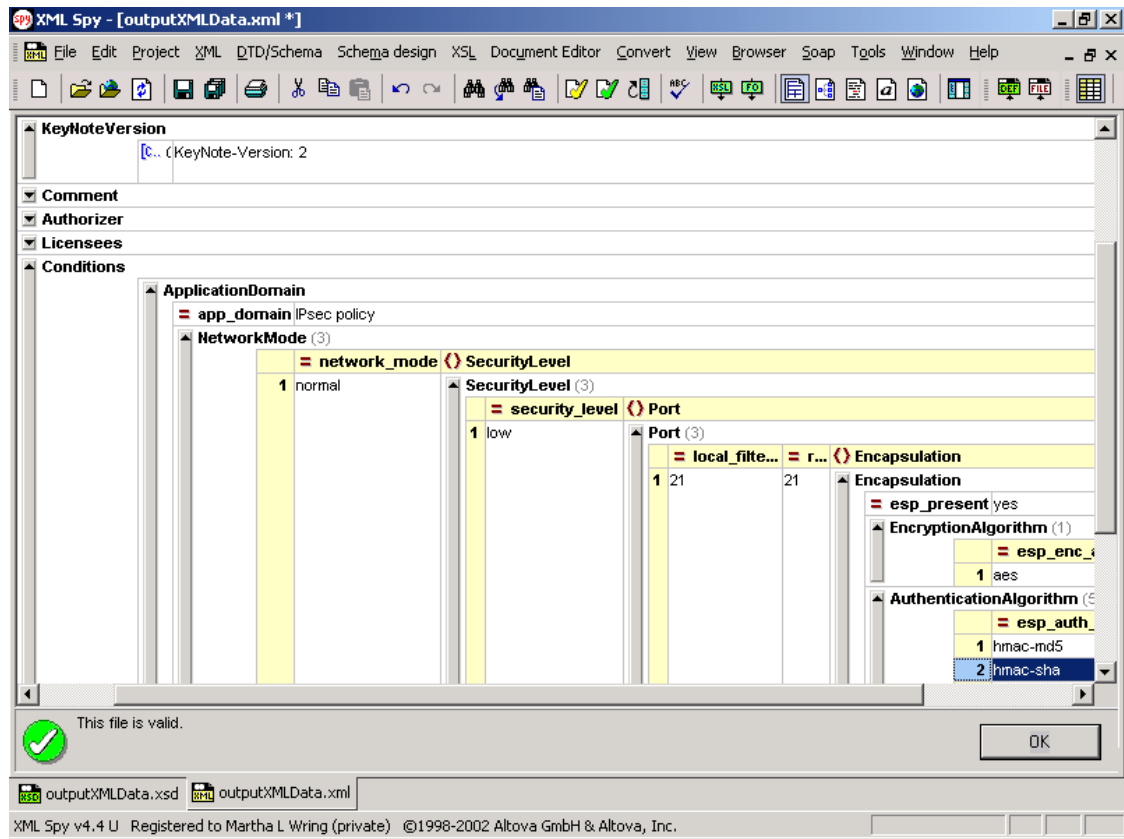
Figure 18.   Editing and Validation of XML Policy File Using XML Spy
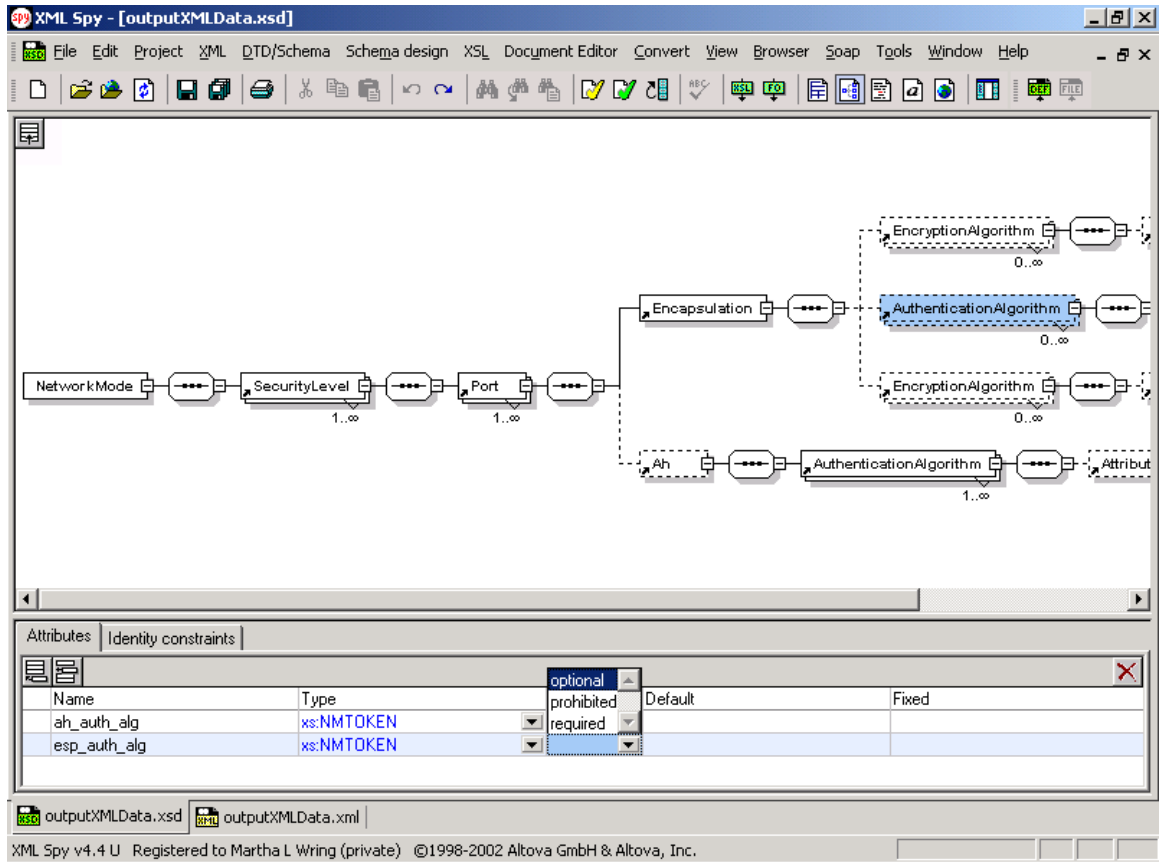
Figure 19.   Schema Design View of the XML Policy Document

The security policy management toolkit thus comprises of the Java based Policy-Editor and an XML editor such as XML Spy, XML Notepad etc. The XML editors are not essential and only aid in manipulating the files and transforming them to multiple forms. The XML editors enable schema validation and also provide tools for manipulation of the XML, Schema and XSL files.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESEARCH SUMMARY AND FUTURE WORK

A. INTRODUCTION

Security policy management is a critical issue in the management of computer and networking resources. IPsec and Keynote provide a mechanism to implement a granular security policy. Previous research in the area of 'Quality of Security Service' demonstrates how an adaptive security policy can provide enhanced security with optimal utilization of network resources. The only missing link in the process is the difficulty in specifying a well-defined, granular, error free and consistent security policy in the language understood by the Keynote trust management engine. This thesis was aimed at finding a solution to this problem by way of developing an easy to use yet powerful security policy editor. The thesis demonstrates that use of XML technology as a middle layer provides us with a means to combine the complexity of Keynote with the simplicity of a policy editor. This novel approach also provides us all the benefits of XML, such as XSL and XML security. While XSL was extensively used, XML security tools could be used as follow up future work.

B. SUMMARY OF RESEARCH PERFORMED IN THIS THESIS

The thesis involved thorough study and implementation of concepts from a wide variety of areas. Extensive programming in Java and integration with the emerging XML technologies was achieved. Concepts from computer security, networking and programming languages were studied and adapted to meet the requirement set forth in the thesis, i.e. security-policy management. A fully functional XML based Policy-Editor was developed and tested. In particular the thesis involved the following:

- Study of IPsec and Keynote Trust Management System,

- Understanding of Quality of Security Service Concepts,

- In depth understanding of different XML technologies,

- Development of human intuitive representation of complex low level policy structures,

49

- Introducing a novel approach to Security Policy management using XML, and

- Extensive Java and XML programming.

C.   FUTURE WORK

Security policy management is a vast area and the research done in this thesis can be complemented with additional work in a number of areas to provide better tools for policy management. Listed below are several major items that will require attention.

1.   Policy File Format

The XML policy file format currently specified could benefit from a more elaborate format with tags for other parameters. XML Namespaces and XML vocabularies could be utilized for a more comprehensive policy format. Different possible combinations of security attributes need to be taken into consideration in the policy specification. Examples of further implementation could involve incorporating other parameters such as algorithm key length, time-of-day parameters etc. As explained in Section IV(C), the policy format should be able to accommodate other Boolean operators such as inequality definitions ($<, >, !=$) in the security policy management mechanism. For example, esp_enc_alg $>$ DES could imply 3DES and AES if we have an ordering for the 'security strength' of each algorithm. Global policy statements such as encryption in crisis mode $<$ 3DES, etc. should be possible. Inclusion of IP addresses in policy statements should also be made possible. The concepts for these have been demonstrated in the implementation and other options analyzed in various chapters. Addition of more parameters as stated above would however open up possibilities for inconsistencies in policy statements and the same will have to be carefully and formally worked out.

2.   Schema Design and RELAX NG

W3C XML Schemas are complicated and hard to formulate. The schema generated in this thesis was automatically generated by XML Spy and modified manually to suit our current requirement. This schema language is very complex and permits us to define complex content models. The schema for instance could be made more specific or more general. This would depend on how we intend to formulate the policy statements.

Future work could therefore focus on how a detailed policy encompassing various parameters such as IP addresses, encryption attributes etc could be specified. The interrelationship between different elements could also be specified, such as if 'Crisis' mode uses encryption then so should 'Normal' and 'Impacted' mode. In exploring more usage of Schema, alternate schema languages such as RELAX NG could be tried. This new language is gaining popularity due its simplicity and robustness. RELAX NG is the result of merging two popular schema languages: RELAX and TREX. The RELAX NG language is very similar to the W3C XML Schema language. RELAX NG has a much stronger foundation in mathematical models, which allows programmers to create highly optimized validation tools. In addition, RELAX NG omits many features that make W3C XML Schemas difficult to learn. RELAX NG, like W3C XML Schemas, is written in an XML Syntax and requires you to define the allowable elements and attributes within your instance documents. RELAX NG can be found on the Web at http://relaxng.org.

3.   Policy Editor Enhancements

The policy editor interface though complete and functional can be improved upon. The particular improvements envisaged are as follows:

o   Data Binding.

Data Binding is a concept where XML data can be read into applications as objects. By accessing the data as Java objects, manipulation becomes faster and managing large policy files would not slow down the system. Through the use of data binding the policy editor could be made more efficient and faster.

o   Global policy settings.

The policy editor could be modified to enable global policy settings. For instance we could have a statement such as all ports should have a minimum encryption of DES or the maximum encryption algorithm for Crisis mode should not exceed 3DES etc. The global settings option could enter the default settings for all permissible ports and then more granular changes could be made

o   Help.

Help to guide the user to form syntactically correct policy statements and correct use of GUI could make the editor more complete. Context sensitive help could also be added

o   Translation and viewing XML.

XML translation and viewing currently need the help of any general purpose XML editor. Using Java packages such as Javax, the same could be incorporated into the GUI thus dispensing the need for XML editors for translation.

o   Schema validation.

Validation of the XML document against DTD and Schema need to be incorporated into the GUI. The same is currently done using an XML tool such as XML Spy. DOM/JDOM/SAX could be used for the purpose.

o   Inconsistency and contradiction checks.

As the policy file is extended to include global parameters and overlapping rules apply to a particular port or application, inconsistencies and contradictions would begin to emerge. The same would have to be considered and avoided. Various XML tools could help in achieving this. Distributed IPsec policy when considered would also give rise to multiple issues of policy consistencies.

o   Improvement in the look and feel of the user interface.

The look and feel of the editor can always be improved to cater for user preferences and to avoid chances of introducing inadvertent errors. Context sensitive tool tips, toolbars and help could all be incorporated into the policy editor to give it a complete look.

3.   XML Interface to Keynote

It is felt that extending the XML policy language specified in this thesis to a broader XML specification and providing an XML processor in the Keynote engine itself would greatly enhance the use of Keynote. This would probably reduce the overhead of parsing in Keynote and provide the power of XML to it for better auditing and dynamic management of trust. XML security features could also be incorporated. For instance

using XML signature, a user can sign for parts of the XML document i.e. a subset of the 'Elements'. Thus making him accountable for the parts signed by him only. By providing an XML interface to Keynote, application users could define their own versions of the policy language and use XSL for translating it into the desired Keynote format, which would be trivial, or alternatively they could use the vocabulary specified in the Keynote specifications.

D.   CONCLUSION

This thesis proves the concept of using XML for management of security policy. A Policy-Editor based on these concepts was designed and developed. Software design and development is ever evolving and this chapter highlights various areas for future work and enhancements. These should eventually lead to complete utilization of the power of IPsec and result in more secure networks.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. XML POLICY FILE

A sample XML Policy file, with various permissible parameters is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U by Raj Mohan) -->
<Policy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <!-- href="C:\XMLProject\currentPolicyXMLXSL\KeynotePolicyModified02JunXSL.xsl" -->
    <KeyNoteVersion><![CDATA[KeyNote-Version: 2
        ]]></KeyNoteVersion>
    <Comment><![CDATA[Comment: Policy file for Network modes and Security Levels in XML Format
        ]]></Comment>
    <Authorizer><![CDATA[Authorizer: "POLICY"
        ]]></Authorizer>
    <Licensees><![CDATA[Licensees: "passphrase:mekmitasdigoat"
        ]]></Licensees>
    <Conditions>
        <ApplicationDomain app_domain="IPsec policy">
            <NetworkMode network_mode="normal">
                <SecurityLevel security_level="low">
                    <Port local_filter_port="21" remote_filter_port="21">
                        <Encapsulation esp_present="yes">
                            <EncryptionAlgorithm esp_enc_alg="des" />
                            <EncryptionAlgorithm esp_enc_alg="des3" />
                        </Encapsulation>
                    </Port>
                    <Port local_filter_port="23" remote_filter_port="23">
                        <Encapsulation esp_present="yes">
                            <EncryptionAlgorithm esp_enc_alg="des" />
                            <EncryptionAlgorithm esp_enc_alg="des-iv64" />
                            <EncryptionAlgorithm esp_enc_alg="des3" />
                            <EncryptionAlgorithm esp_enc_alg="idea3" />
                            <EncryptionAlgorithm esp_enc_alg="des-iv32" />
                            <EncryptionAlgorithm esp_enc_alg="aes" />
                        </Encapsulation>
                        <Ah ah_present="yes">
                            <AuthenticationAlgorithm ah_auth_alg="hmac-sha" />
                            <AuthenticationAlgorithm ah_auth_alg="des-mac" />
                        </Ah>
                    </Port>
                    <Port local_filter_port="100" remote_filter_port="100">
                        <Encapsulation esp_present="yes">
                            <EncryptionAlgorithm esp_enc_alg="des" />
                            <EncryptionAlgorithm esp_enc_alg="des-iv64" />
                            <EncryptionAlgorithm esp_enc_alg="des3" />
                            <EncryptionAlgorithm esp_enc_alg="des-iv32" />
                            <AuthenticationAlgorithm esp_auth_alg="hmac-md5" />
                            <AuthenticationAlgorithm esp_auth_alg="hmac-sha" />
                        </Encapsulation>
                        <Ah ah_present="yes">
                            <AuthenticationAlgorithm ah_auth_alg="hmac-md5" />
                            <AuthenticationAlgorithm ah_auth_alg="hmac-sha" />
                        </Ah>
                    </Port>
                </SecurityLevel>
                <SecurityLevel security_level="medium">
                    <Port local_filter_port="21" remote_filter_port="21">
                        <Encapsulation esp_present="yes">
                            <EncryptionAlgorithm esp_enc_alg="des" />
                            <EncryptionAlgorithm esp_enc_alg="des3" />
                            <EncryptionAlgorithm esp_enc_alg="des-iv32" />
```

```xml
                </Encapsulation>
              </Port>
          </SecurityLevel>
          <SecurityLevel security_level="high" />
        </NetworkMode>
        <NetworkMode network_mode="impacted">
            <SecurityLevel security_level="low" />
            <SecurityLevel security_level="medium" />
            <SecurityLevel security_level="high" />
        </NetworkMode>
        <NetworkMode network_mode="crisis">
            <SecurityLevel security_level="low" />
            <SecurityLevel security_level="medium" />
            <SecurityLevel security_level="high" />
        </NetworkMode>
      </ApplicationDomain>
    </Conditions>
    <Dummy><![CDATA[
        ]]></Dummy>
</Policy>
```

APPENDIX B. KEYNOTE POLICY TEMPLATE

The following is the XSL stylesheet that transforms the XML Policy file into the KeyNote Policy file syntax.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="text" version="1.0" encoding="UTF-8" indent="yes"/>
        <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
        <!-- root template -->
        <!-- apply the templates for all the children of the /policy node -->
        <!-- just call for each child .. hence can add more children to the source tree and only need to add the
corresponding template here or let it use the default template -->
        <xsl:template match="/">
                <xsl:apply-templates/>
        </xsl:template>
        <!-- root template -->
        <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
        <!-- GENERIC TOP LEVEL  TEMPLATE  ****************************************-->
        <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
        <!-- generic  template for all other children of policy -->
        <xsl:template match="/Policy/*" priority=".8">
                <xsl:value-of select="."/>
        </xsl:template>
        <!-- children of policy  template -->
        <!-- ~~~~~~~~~~~~~~~~~~~~~ -->
        <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
        <!-- CONDITIONS TEMPLATE  ********************************************** -->
        <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
        <!-- conditions  template -->
        <xsl:template match="/Policy/Conditions" priority=" 0.5">
                <!-- start the opening brackets for the conditions  clause -->
                <xsl:text>Conditions: ( </xsl:text>
                <!-- Handle the ApplicationDomain element .. print its attribute and values in a bracket -->
                <xsl:for-each select="child::ApplicationDomain">
                        <xsl:call-template name="oneAttributeContent"/>
                </xsl:for-each>
                <!-- for each network mode, open a bracket enter the attribute name = value close  -->
                <!-- the bracket and place the operator && and then call the template for the level children -->
                <!-- after the call close the bracket -->
                <!-- if the node is not the first node or if it had precedent-siblings which were also of -->
                <!-- network node type then place an || before starting the first bracket -->
                <!-- || ( (network_mode == normal) && -->
                <xsl:for-each select="ApplicationDomain/NetworkMode">
                        <!-- first network node to be handled here -->
                        <xsl:if test="(count(preceding-sibling::NetworkMode) = 0)">
                                <xsl:text> &amp;&amp; </xsl:text>
                                <xsl:text> ( </xsl:text>
                                <xsl:call-template name="oneAttributeContent"/>
                                <!-- handle all children of network node here -->
                                <xsl:for-each select="SecurityLevel">
                                        <!-- first SecurityLevel node to be handled here -->
                                        <xsl:if test="(count(preceding-sibling::SecurityLevel) = 0)">
                                                <xsl:text> &amp;&amp; </xsl:text>
                                                <xsl:text> ( </xsl:text>
                                                <xsl:call-template name="oneAttributeContent"/>
                                                <!-- handle children of SecurityLevel here -->
                                                <!-- children here -->
                                        </xsl:if>
                                        <!--end if securtiyLevel node is the first one -->
                                        <!-- except first case preceed with an or symbol -->
                                        <xsl:if test="(count(preceding-sibling::SecurityLevel) > 0)">
                                                <xsl:text> || </xsl:text>
                                                <xsl:text> ( </xsl:text>
                                                <xsl:call-template name="oneAttributeContent"/>
```

```xml
                                        <!-- handle children of SecurityLevel here -->
                                        <!-- children here -->
                        </xsl:if>
                        <!-- last SecurityLevel node to be handled here close bracket -->
                        <xsl:if test="(count(following-sibling::SecurityLevel) = 0)"/>
                        <xsl:text> ) </xsl:text>
                </xsl:for-each>
                <!-- for all levels -->
        </xsl:if>
        <!-- if network node is the first one -->
        <!-- except first case preceed with an or symbol -->
        <xsl:if test="(count(preceding-sibling::NetworkMode) > 0)">
                        <xsl:text> || </xsl:text>
                        <xsl:text> ( </xsl:text>
                        <xsl:call-template name="oneAttributeContent"/>
                        <!-- handle all children of network node here -->
                        <xsl:for-each select="SecurityLevel">
                                <!-- first SecurityLevel node to be handled here -->
                                <xsl:if test="(count(preceding-sibling::SecurityLevel) = 0)">
                                        <xsl:text> &amp;&amp; </xsl:text>
                                        <xsl:text> ( </xsl:text>
                                        <xsl:call-template name="oneAttributeContent"/>
                                        <!-- handle children of SecurityLevel here -->
                                        <!-- children here -->
                                </xsl:if>
                                <!--end if securtiyLevel node is the first one -->
                                <!-- except first case preceed with an or symbol -->
                                <xsl:if test="(count(preceding-sibling::SecurityLevel) > 0)">
                                        <xsl:text> || </xsl:text>
                                        <xsl:text> ( </xsl:text>
                                        <xsl:call-template name="oneAttributeContent"/>
                                        <!-- handle children of SecurityLevel here -->
                                        <!-- children here -->
                                </xsl:if>
                                <!-- last SecurityLevel node to be handled here close bracket -->
                                <xsl:if test="(count(following-sibling::SecurityLevel) = 0)"/>
                                <xsl:text> ) </xsl:text>
                        </xsl:for-each>
                        <!-- for all levels -->
        </xsl:if>
        <!-- last network node to be handled here close bracket -->
        <xsl:if test="(count(following-sibling::NetworkMode) = 0)"/>
        <xsl:text> ) </xsl:text>
</xsl:for-each>
<!-- end network nodes -->
<xsl:text> ) </xsl:text>
</xsl:template>
<!-- conditions  template -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- CONDITIONS TEMPLATE  ********************************************** -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- conditions  template -->
<xsl:template match="/Policy/Conditions" priority="1">
        <!-- start the opening brackets for the conditions  clause -->
        <xsl:text>Conditions: </xsl:text>
        <!-- Handle the ApplicationDomain element .. print its attribute and values in a bracket -->
        <xsl:for-each select="child::ApplicationDomain">
                <xsl:call-template name="appDomainContent"/>
        </xsl:for-each>
        <xsl:text>-> "true";</xsl:text>
</xsl:template>
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!--  NAMED TEMPLATE FOR APPLICATION DOMAIN   ********************************* -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- ( (app_domain == "IPsec policy") && ( (netwokmode1) || (networkmode2) ... ) ) -->
<xsl:template name="appDomainContent">
        <xsl:text> ( </xsl:text>
        <xsl:call-template name="oneAttributeContent"/>
        <xsl:for-each select="NetworkMode">
```

58

```xml
                    <xsl:if test="(count(preceding-sibling::NetworkMode) = 0)">
                            <xsl:text> &amp;&amp; </xsl:text>
                            <xsl:text> ( </xsl:text>
                            <xsl:call-template name="networkModeContent"/>
                    </xsl:if>
                    <!--end if NetworkMode node is the first one -->
                    <!-- except first case preceed with an or symbol -->
                    <xsl:if test="(count(preceding-sibling::NetworkMode) > 0)">
                            <xsl:text> || </xsl:text>
                            <xsl:call-template name="networkModeContent"/>
                    </xsl:if>
                    <!-- last NetworkModel node to be handled here close bracket -->
                    <xsl:if test="(count(following-sibling::NetworkMode) = 0)">
                            <xsl:text> ) </xsl:text>
                    </xsl:if>
            </xsl:for-each>
            <!-- end for each NetworkMode -->
            <xsl:text> ) </xsl:text>
</xsl:template>
<!-- end appDomainContent template -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!--  NAMED TEMPLATE FOR NETWORK MODE  ********************************* -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- ( (network_mode == "normal") && ( (security1) || (seucurity2) ... ) ) -->
<xsl:template name="networkModeContent">
            <xsl:text> ( </xsl:text>
            <xsl:call-template name="oneAttributeContent"/>
            <xsl:for-each select="SecurityLevel">
                    <xsl:if test="(count(preceding-sibling::SecurityLevel) = 0)">
                            <xsl:text> &amp;&amp; </xsl:text>
                            <xsl:text> ( </xsl:text>
                            <xsl:call-template name="securityContent"/>
                    </xsl:if>
                    <!--end if SecurityLevel node is the first one -->
                    <!-- except first case preceed with an or symbol -->
                    <xsl:if test="(count(preceding-sibling::SecurityLevel) > 0)">
                            <xsl:text> || </xsl:text>
                            <xsl:call-template name="securityContent"/>
                    </xsl:if>
                    <!-- last securityLevel node to be handled here close bracket -->
                    <xsl:if test="(count(following-sibling::SecurityLevel) = 0)">
                            <xsl:text> ) </xsl:text>
                    </xsl:if>
            </xsl:for-each>
            <!-- end for each securityLevel -->
            <xsl:text> ) </xsl:text>
</xsl:template>
<!-- end network mode template -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!--  NAMED TEMPLATE FOR SECURITY  ********************************* -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- ( (security_level == "low") && ( (port1) || (port2) ... ) ) -->
<xsl:template name="securityContent">
            <xsl:text> ( </xsl:text>
            <xsl:call-template name="oneAttributeContent"/>
            <xsl:for-each select="Port">
                    <xsl:if test="(count(preceding-sibling::Port) = 0)">
                            <xsl:text> &amp;&amp; </xsl:text>
                            <xsl:text> ( </xsl:text>
                            <xsl:call-template name="portContent"/>
                    </xsl:if>
                    <!--end if Port node is the first one -->
                    <!-- except first case preceed with an or symbol -->
                    <xsl:if test="(count(preceding-sibling::Port) > 0)">
                            <xsl:text> || </xsl:text>
                            <xsl:call-template name="portContent"/>
                    </xsl:if>
                    <!-- last Port node to be handled here close bracket -->
                    <xsl:if test="(count(following-sibling::Port) = 0)">
                            <xsl:text> ) </xsl:text>
```

```
                        </xsl:if>
                </xsl:for-each>
                <!-- end for each Port -->
                <xsl:text> ) </xsl:text>
        </xsl:template>
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- NAMED TEMPLATE FOR PORT  ********************************** -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- outcome : take the two attributes local port num and remote port number and or them in a bracket -->
        <!-- and and it with the templates of its children i.ed encapsulation and Ah ored -->
        <!-- e.g ( ( ( local_port == "23") || (remote_port == "23") ) && ( (encapsulation) || (authentication) ) ) -->
        <xsl:template name="portContent">
                <xsl:text> ( </xsl:text>
                <xsl:call-template name="portTwoAttributeContent"/>
                <xsl:text> &amp;&amp; </xsl:text>
                <xsl:text> ( </xsl:text>
                <xsl:for-each select="Encapsulation">
                        <!-- Encapsulation node to be handled here -->
                        <xsl:call-template name="encapsulationOrAh"/>
                </xsl:for-each>
                <!-- for all encryption algorithms  -->
                <xsl:for-each select="Ah">
                        <!--Ah node to be handled here -->
                        <xsl:if test="(count(preceding-sibling::Encapsulation) > 0)">
                                <xsl:text> || </xsl:text>
                        </xsl:if>
                        <xsl:call-template name="encapsulationOrAh"/>
                </xsl:for-each>
                <xsl:text> ) </xsl:text>
                <xsl:text> ) </xsl:text>
        </xsl:template>
        <!-- port  template -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- GENERIC NAMED TWO ATTRIBUTE TEMPLATE  ********************************** -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- generic named template will print the attrbute and its value inside a bracket -->
        <!-- TODO need to check if it has more than needed attribute then only apply to the first one -->
        <!-- eg output ((local_filter_port == "23" ) || (local_filter_port == "23" ))   of the context nodes attribute -->
        <xsl:template name="portTwoAttributeContent" priority=".8">
                <xsl:text>( </xsl:text>
                <xsl:for-each select="@local_filter_port">
                        <xsl:text>( </xsl:text>
                        <xsl:value-of select="local-name()"/>
                        <xsl:text> == </xsl:text>
                        <xsl:text>"</xsl:text>
                        <xsl:value-of select="."/>
                        <xsl:text>"</xsl:text>
                        <xsl:text> ) </xsl:text>
                </xsl:for-each>
                <xsl:text> || </xsl:text>
                <xsl:for-each select="@remote_filter_port">
                        <xsl:text>( </xsl:text>
                        <xsl:value-of select="local-name()"/>
                        <xsl:text> == </xsl:text>
                        <xsl:text>"</xsl:text>
                        <xsl:value-of select="."/>
                        <xsl:text>"</xsl:text>
                        <xsl:text> ) </xsl:text>
                </xsl:for-each>
                <xsl:text>) </xsl:text>
        </xsl:template>
        <!-- children of policy  template -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- NAMED TEMPLATE FOR ENCAPSULATION OR AH ********************************** -->
        <!-- ∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼∼ -->
        <!-- outcome should be ( (esp_present == "yes") && ( .. call for each encryption algo.. with ors in between then
&& then call for auth and close ) -->
        <!-- ( (esp_present == "yes") && | ( (enc1) || (enc2) ) && ((ah1)||(ah2)) | ) -->
```

60

```
<!-- TODO check for inconsistency ... ie esp_present == yes but no enc alg specified -->
<xsl:template name="encapsulationOrAh">
        <xsl:text> ( </xsl:text>
        <xsl:call-template name="oneAttributeContent"/>
        <xsl:text> &amp;&amp; </xsl:text>
        <xsl:text> (</xsl:text>
        <!-- open a ( and call for each algorithm where type = encryption and put ors in between and close
with ) -->

        <xsl:for-each select="EncryptionAlgorithm">
                <!-- first encryption algorithm node to be handled here -->
                <xsl:if test="(count(preceding-sibling::EncryptionAlgorithm) = 0)">
                        <xsl:text>( </xsl:text>
                        <xsl:call-template name="algorithm"/>
                </xsl:if>
                <!--end if encryptionnode is the first one -->
                <!-- except first case preceed with an or symbol -->
                <xsl:if test="(count(preceding-sibling::EncryptionAlgorithm) > 0)">
                        <xsl:text> || </xsl:text>
                        <xsl:call-template name="algorithm"/>
                </xsl:if>
                <!-- last encryption algorithm  node to be handled here close bracket -->
                <xsl:if test="(count(following-sibling::EncryptionAlgorithm) = 0)">
                        <xsl:text>) </xsl:text>
                </xsl:if>
        </xsl:for-each>
        <!-- for all encryption algorithms  -->
        <!-- insert && between encryption and auth types-->
        <xsl:if test="(count(child::EncryptionAlgorithm) > 0)">
                <xsl:text> &amp;&amp; </xsl:text>
        </xsl:if>
        <!-- open a ( and call for each algorithm where type = authentication and put ors in between and
close with ) -->

        <xsl:for-each select="AuthenticationAlgorithm">
                <!-- first authentication algorithm node to be handled here -->
                <xsl:if test="(count(preceding-sibling::AuthenticationAlgorithm) = 0)">
                        <xsl:text>( </xsl:text>
                        <xsl:call-template name="algorithm"/>
                </xsl:if>
                <!--end if encryptionnode is the first one -->
                <!-- except first case preceed with an or symbol -->
                <xsl:if test="(count(preceding-sibling::AuthenticationAlgorithm) > 0)">
                        <xsl:text> || </xsl:text>
                        <xsl:call-template name="algorithm"/>
                </xsl:if>
                <!-- last authentication algorithm  node to be handled here close bracket -->
                <xsl:if test="(count(following-sibling::AuthenticationAlgorithm) = 0)">
                        <xsl:text>) </xsl:text>
                </xsl:if>
        </xsl:for-each>
        <!-- for all authentication algorithms  -->
        <!-- final closing of all the brackets -->
        <xsl:text> ) </xsl:text>
        <xsl:text> ) </xsl:text>
</xsl:template>
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!--  NAMED TEMPLATE FOR ALGORITHM AND ITS ATTRIBUTES***************** -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- outcome when called in the context of encryptionAlgorithm node -->
<!-- ( (esp_enc_alg == "DES)&& ( (esp_key_length == "128) )( (esp_key_length == "128) .. ) ) -->
<!-- caller need not enclose these in a bracket  -->
<!-- call one such for each algorithm -->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<!-- template forEncryptionAlgorithm -->
<xsl:template name="algorithm" priority=".8">
        <xsl:text> ( </xsl:text>
        <xsl:call-template name="oneAttributeContent"/>
        <xsl:for-each select="Attribute">

                                        <xsl:if test="(count(preceding-sibling::Attribute) = 0)">
                                                <xsl:text> &amp;&amp; </xsl:text>
```

```xml
                                        <xsl:text> ( </xsl:text>
                                        <xsl:call-template name="oneAttributeContent"/>
                            </xsl:if>

                            <!--end if Attribute node is the first one -->
                            <!-- except first case preceed with an or symbol -->
                            <xsl:if test="(count(preceding-sibling::Attribute) > 0)">
                                        <xsl:text> || </xsl:text>
                                        <xsl:call-template name="oneAttributeContent"/>
                            </xsl:if>
                            <!-- last Attributel node to be handled here close bracket -->
                            <xsl:if test="(count(following-sibling::Attribute) = 0)">
                                        <xsl:text> ) </xsl:text>


                            </xsl:if>
                </xsl:for-each>
                <!-- end for each attribute -->

                <xsl:text> ) </xsl:text>
    </xsl:template>
    <!-- children of policy  template -->
    <!-- ~~~~~~~~~~~~~~~~~~~ -->
    <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <!-- GENERIC NAMED ONE ATTRIBUTE TEMPLATE  ********************* -->
    <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
    <!-- generic named template will print the attrbute and its value inside a bracket -->
    <!-- TODO need to check if it has more than one attribute then only apply to the first one -->
    <!-- currently lists all the attributes and put an == and the value and enclose the complete thing in brackets -->
    <!-- eg output (attributeName == "value" )  of the context nodes attribute -->
    <xsl:template name="oneAttributeContent" priority=".8">
                <xsl:for-each select="@*">
                            <xsl:text>( </xsl:text>
                            <xsl:value-of select="local-name()"/>
                            <xsl:text> == </xsl:text>
                            <xsl:text>"</xsl:text>
                            <xsl:value-of select="."/>
                            <xsl:text>"</xsl:text>
                            <xsl:text> ) </xsl:text>
                </xsl:for-each>
    </xsl:template>
    <!-- children of policy  template -->
    <!-- ~~~~~~~~~~~~~~~~~~~ -->
</xsl:stylesheet>
```

APPENDIX C. WEB TEMPLATE

The following is the XSL stylesheet that transforms the XML Policy file into a web page, depicting the policy settings in tables and color.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:template match="/">
        <html>
            <head>
                <title>
                    Security Policy State

                </title>
            </head>
            <body>
                <h4 align="center">
                    <a name="DocumentTop"/>
                    <u>
                        <ul/>
                        <h1>Security Policy State</h1>
                    </u>
                </h4>
                <u>
                    <ul/>
                </u>
                <h4>Jump To:</h4>
                <xsl:for-each select="Policy/Conditions/ApplicationDomain/NetworkMode">
                    <xsl:for-each select="SecurityLevel">
                        <a href="{concat('#', ../@network_mode, @security_level)}">
                            <xsl:value-of select="../@network_mode"/>
                            <xsl:text> - </xsl:text>
                            <xsl:value-of select="@security_level"/>
                        </a>
                        <br/>
                    </xsl:for-each>
                </xsl:for-each>
                <xsl:for-each select="Policy/Conditions/ApplicationDomain/NetworkMode">
                    <xsl:for-each select="SecurityLevel">
                        <!-- Table goes in here -->
                        <table border="1" bgcolor="000000" title="StatusTableTitle" align="center">
                            <caption title="Our TableCaption@title">
                            <h3> Security State of Network Policy : <a name="{concat(../@network_mode, @security_level)}">
                                    <xsl:value-of select="../@network_mode"/>
                                    <xsl:text> - </xsl:text>
                                    <xsl:value-of select="@security_level"/>
                                </a>
                                <br/>
                                Number of Ports Open: <xsl:value-of select="count(Port)"/>
                            </h3>
                            </caption>
                            <thead align="center" valign="middle">
                                <!--Header-->
                            </thead>
                            <tbody>
                                <tr>
                                    <!-- row 1 -->
                                    <!-- row 1headers (1 coln)-->
                                    <th rowspan="2" bgcolor="cccccc">Port No</th>
                                    <!-- row 1 Encryption headers (5 colns) -->
                                    <th colspan="12" bgcolor="aa9966">Encryption Types</th>
                                    <!-- row 1 Authentication headers (3 Colns)-->
                                    <th bgcolor="lightblue" colspan="5">AuthenticationTypes</th>
                                </tr>
                                <tr>
                                    <!-- row 2 -->
```

63

```
                        <!-- row 2 Encryption headers (5 colns) -->
                        <th bgcolor="cc9966">DES</th>
                        <th bgcolor="cc9966">DES-IV64</th>
                        <th bgcolor="cc9966">DES-IV32</th>
                        <th bgcolor="cc9966">3DES</th>
                        <th bgcolor="cc9966">RC4</th>
                        <th bgcolor="cc9966">RC5</th>
                        <th bgcolor="cc9966">AES</th>
                        <th bgcolor="cc9966">IDEA</th>
                        <th bgcolor="cc9966">3IDEA</th>
                        <th bgcolor="cc9966">BLOWFISH </th>
                        <th bgcolor="cc9966">CAST</th>
                        <th bgcolor="cc9966">None</th>
                        <!-- row 2 Authentication headers (3 Colns)-->
                        <th bgcolor="0066ff">HMAC-MD5</th>
                        <th bgcolor="0066ff">HMAC-SHA-1</th>
                        <th bgcolor="0066ff">HMAC-RIPEMD</th>
                        <th bgcolor="0066ff">KPDK</th>
                        <th bgcolor="0066ff">None</th>
            </tr>
            <!-- rows for each port -->
            <xsl:for-each select="Port">
                <tr>
                        <th bgcolor="yellow">
                                <h4>
                                        <xsl:value-of select="@local_filter_port"/>
                                </h4>
                        </th>
                        <!--for each port, color appropriate cell in table-->
                        <xsl:choose>
                                <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'des' ">
                                        <td BGCOLOR="#00ff00"/>
                                </xsl:when>
                                <xsl:otherwise>
                                        <td BGCOLOR="#ffffff"/>
                                </xsl:otherwise>
                        </xsl:choose>
                        <xsl:choose>
                                <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'des-iv64' ">
                                        <td BGCOLOR="#00ff00"/>
                                </xsl:when>
                                <xsl:otherwise>
                                        <td BGCOLOR="#ffffff"/>
                                </xsl:otherwise>
                        </xsl:choose>
                        <xsl:choose>
                                <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'des-iv32' ">
                                        <td BGCOLOR="#00ff00"/>
                                </xsl:when>
                                <xsl:otherwise>
                                        <td BGCOLOR="#ffffff"/>
                                </xsl:otherwise>
                        </xsl:choose>
                        <xsl:choose>
                                <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'des3' ">
                                        <td BGCOLOR="#00ff00"/>
                                </xsl:when>
                                <xsl:otherwise>
                                        <td BGCOLOR="#ffffff"/>
                                </xsl:otherwise>
                        </xsl:choose>
                        <xsl:choose>
                                <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'rc4' ">
                                        <td BGCOLOR="#00ff00"/>
                                </xsl:when>
                                <xsl:otherwise>
                                        <td BGCOLOR="#ffffff"/>
                                </xsl:otherwise>
                        </xsl:choose>
                        <xsl:choose>
```

64

```xml
            <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'rc5' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'aes' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'idea' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'idea3' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
        <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'blowfish' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'cast' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <xsl:when test="Encapsulation/EncryptionAlgorithm/@esp_enc_alg = 'none' ">
                <td BGCOLOR="#ff0000"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
<xsl:when test="Encapsulation/AuthenticationAlgorithm/@esp_auth_alg = 'hmac-md5' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
<xsl:when test="Encapsulation/AuthenticationAlgorithm/@esp_auth_alg = 'hmac-sha' ">
                <td BGCOLOR="#00ff00"/>
            </xsl:when>
            <xsl:otherwise>
                <td BGCOLOR="#ffffff"/>
            </xsl:otherwise>
```

```xml
                    </xsl:choose>
                    <xsl:choose>
            <xsl:when test="Encapsulation/AuthenticationAlgorithm/@esp_auth_alg = 'hmac-ripemd' ">
                        <td BGCOLOR="#00ff00"/>
                </xsl:when>
                <xsl:otherwise>
                    <td BGCOLOR="#ffffff"/>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:choose>
            <xsl:when test="Encapsulation/AuthenticationAlgorithm/@esp_auth_alg = 'kpdk' ">
                        <td BGCOLOR="#00ff00"/>
                </xsl:when>
                <xsl:otherwise>
                    <td BGCOLOR="#ffffff"/>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:choose>
            <xsl:when test="Encapsulation/AuthenticationAlgorithm/@esp_auth_alg = 'none' ">
                        <td BGCOLOR="#ff0000"/>
                </xsl:when>
                <xsl:otherwise>
                    <td BGCOLOR="#ffffff"/>
                </xsl:otherwise>
            </xsl:choose>
                </tr>
            </xsl:for-each>
            </tbody>
        </table>
    <!-- for each port, list the esp encryption algorithms, esp authentication algorithms and ah authentication
    algorithms -->
        <xsl:for-each select="Port">
            <h4>Port:  <xsl:value-of select="@local_filter_port"/>
            </h4>
            esp Encryption Algorithm(s):

            <xsl:for-each select="Encapsulation/EncryptionAlgorithm">
                <xsl:value-of select="@esp_enc_alg"/>
                <xsl:text/>
            </xsl:for-each>
            <br/>

            esp Authenication Algorithm(s):

            <xsl:for-each select="Encapsulation/AuthenticationAlgorithm">
                <xsl:value-of select="@esp_auth_alg"/>
                <xsl:text/>
            </xsl:for-each>
            <br/>

            ah Authentication Algorithm(s):
            <xsl:for-each select="Ah/AuthenticationAlgorithm">
                <xsl:value-of select="@ah_auth_alg"/>
                <xsl:text/>
            </xsl:for-each>
            <br/>
            <br/>
        </xsl:for-each>
        <h5 align="center">
            <a href="#DocumentTop">Back to Top of Document</a>
        </h5>
        <hr/>
        </xsl:for-each>
        </xsl:for-each>
        </body>
    </html>
    </xsl:template>
</xsl:stylesheet>
```

APPENDIX D. ISAKMPD.POLICY FILE

The following is a sample of the **isakmpd.policy** file (From Agar, December 2001). This is the format that the XML policy file (User Policy File) is transformed into by the stylesheet in Appendix B

```
KeyNote-Version: 2
Comment: Policy file for Network Modes and Security Levels
Authorizer: "POLICY"
Licensees: "passphrase:mekmitasdigoat"
Conditions: ( (app_domain == "IPsec policy") &&
        (
          ( (network_mode == "normal") &&
           (
             ( (security_level == "low") &&
              (
                ( (esp_present == "yes") &&
                  ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
                  (esp_enc_alg == "des") &&
                  (esp_auth_alg == "hmac-md5")
                )
                ||
                ( (ah_present == "yes") &&
                  ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
                  (ah_auth_alg == "hmac-md5")
                )
              )
            )
            ||
            ( (security_level == "medium") &&
             (
               ( (esp_present == "yes") &&
                 ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
                 (esp_enc_alg == "cast") &&
                 (esp_auth_alg == "hmac-sha")
               )
               ||
               ( (ah_present == "yes") &&
                 ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
                 (ah_auth_alg == "hmac-md5")
               )
             )
            )
            ||
            ( (security_level == "high") &&
```

```
    (
      ( (esp_present == "yes") &&
        ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
        (esp_enc_alg == "3des") &&
        (esp_auth_alg == "hmac-sha")
      )
      ||
      ( (ah_present == "yes") &&
        ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
        (ah_auth_alg == "hmac-sha")
      )
    )
  )
  )
)
||
( (network_mode == "impacted") &&
  (
    ( (security_level == "low") &&
      (
        ( (esp_present == "yes") &&
          ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
          (esp_enc_alg == "des") &&
          (esp_auth_alg == "hmac-md5")
        )
        ||
        ( (ah_present == "yes") &&
          ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
          (ah_auth_alg == "hmac-md5")
        )
      )
    )
    ||
    ( (security_level == "medium") &&
      (
        ( (esp_present == "yes") &&
          ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
          (esp_enc_alg == "des") &&
          (esp_auth_alg == "hmac-md5")
        )
        ||
        ( (ah_present == "yes") &&
          ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
          (ah_auth_alg == "hmac-md5")
        )
      )
    )
```

```
    )
    ||
    ( (security_level == "high") &&
     (
       ( (esp_present == "yes") &&
         ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
         (esp_enc_alg == "3des") &&
         (esp_auth_alg == "hmac-md5")
       )
       ||
       ( (ah_present == "yes") &&
         ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
         (ah_auth_alg == "hmac-sha")
       )
     )
    )
   )
  )
  ||
  ( (network_mode == "crisis") &&
   (
     ( (security_level == "low") &&
      (
        ( (esp_present == "yes") &&
          ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
          (esp_enc_alg == "3des") &&
          (esp_auth_alg == "hmac-sha")
        )
        ||
        ( (ah_present == "yes") &&
          ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
          (ah_auth_alg == "hmac-sha")
        )
      )
     )
     ||
     ( (security_level == "medium") &&
      (
        ( (esp_present == "yes") &&
          ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
          (esp_enc_alg == "3des") &&
          (esp_auth_alg == "hmac-sha")
        )
        ||
        ( (ah_present == "yes") &&
          ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
```

```
            (ah_auth_alg == "hmac-sha")
          )
        )
      )
      ||
      ( (security_level == "high") &&
        (
          ( (esp_present == "yes") &&
            ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
            (esp_enc_alg == "aes") &&
            (esp_auth_alg == "hmac-sha")
          )
          ||
          ( (ah_present == "yes") &&
            ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
            (ah_auth_alg == "hmac-sha")
          )
        )
      )
    )
  )
  ||
  ( (network_mode == "default") &&
    (security_level == "default") &&
    (
      ( (esp_present == "yes") &&
        ( (local_filter_port == "23") || (remote_filter_port == "23") ) &&
        (esp_enc_alg == "des") &&
        (esp_auth_alg == "hmac-md5")
      )
      ||
      ( (ah_present == "yes") &&
        ( (local_filter_port == "79") || (remote_filter_port == "79") ) &&
        (ah_auth_alg == "hmac-md5")
      )
    )
  )
)
-> "true";
```

LIST OF REFERENCES

Agar Christopher, *Dynamic Parameterization of IPsec,* Master of Science Thesis, Department of Computer Science, Naval Postgraduate School, December 2001.

Alfred V. Aho, John E. Hopcroft und Jeffrey D. Ullman. Data Structures and Algorithms. Addison-Wesley Verlag, 1987.

Angelos D. Keromytis, John Ioannidis, and Jonathan M. Smith, *Implementing IPsec*, In Proceedings of the IEEE *Global Internet (GlobeCom) 1997,* pp. 1948 - 1952. November 1997, Phoenix, AZ.

Blaze Matt, Feigenbaum, Joan, Ioannidis, John, and Keromytis, Angelos D, *The KeyNote Trust Management System Version 2, (RFC 2704,* Network Working Group, September 1999, http://www.ietf.org/rfc/rfc2404.txt

Blaze, Matt, Feigenbaum, Joan, and Keromytis, Angelos D., *KeyNote: Trust Management for Public-Key Infrastructures,* In Proceedings of the *1998 Security Protocols International Workshop,* Springer LNCS vol. 1550, pp. 59 - 63. April 1998, Cambridge, England. Also *AT&T Technical Report 98.11.1.*

Blaze, Matt, Ioannidis, John and Keromytis, Angelos D. *Trust Management and Network Security Protocols*, In Proceedings of the *1999 Security Protocols International Workshop,* April 1999, Cambridge, England.

Blaze, Matt, Ioannidis, John and Keromytis, Angelos D., *Trust Management for IPsec,* In Proceedings of the Internet Society *Symposium on Network and Distributed Systems Security (SNDSS) 2001,* pp. 139 - 151. February 2001, San Diego, CA.

David Hunter, Kurt Cagle, Chris Dix, Roger Kovack, Jonathan Pinnock, Jeff Rafter *Beginning XML 2nd Edition* , Wrox Press Ltd,2002.

Doraswamy, Naganand and Harkins, Dan, *IPsec The New Security Standard for the Internet, Intranets, and the Virtual Private Networks*, Prentice-Hall, Inc., 1999.

IETF (Internet Engineering Task Force) RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*, eds. T. Berners-Lee, R. Fielding, L. Masinter. August 1998.*

Irvine, C.E. and Levin, T., *Quality of Security Service, Proceedings of the New Security Paradigms Workshop*, Cork, Ireland, September 2000

Java 2 Standard Edition, V1.2.2 API Specification, http://java.sun.com/products/jdk/1.2/docs/api/, Sun Microsystems, Inc., 1999.

Kent, S and Atkinson, R, *Security Architecture* for the Internet Protocol, RFC2401, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2401.txt

Leiseboer, John, *IPSEC – Security Architecture for IP, Part 2: Security Association*, http://www.chipcenter.com/eexpert/jleiseboer/jleiseboer036.html, ChipCenter: The Web's Definitive Electronics Resource, Modified 12/05/2001.

Maughan, D., Schertler, M., Schneider M., Turner J., *Internet Security Association and Key Management Protocol (ISAKMP), RFC 2408,* Network Working Group*,* November 1998*,* http://www.ietf.org/rfc/rfc2409.txt

Naval Postgraduate School, NPS-CS-02-001, *IPsec Modulation for the Quality of Security Service,* Spyropoulou, Evdoxia., Agar, Christopher D., Levin, Timothy, and Irvine, Cynthia, January 2002.

Naval Postgraduate School, NPS-CS-02-002, *KeyNote Policy Files and Conversion to Disjunctive Normal Form for Use in IPsec,* Spyropoulou, Evdoxia., Levin, Timothy, and Irvine, Cynthia, January 2002.

Naval Postgraduate School, NPS-CS-02-003, *Demonstration of Quality of Security Service Awareness for IPsec,* Spyropoulou, Evdoxia., Levin, Timothy, and Irvine, Cynthia, January 2002.

Thayer, R., Doraswamy,N., and Glenn, R., *IP Security Document Roadmap,* RFC 2411, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2411.txt

Using IPsec *(Internet Security Protocol),* http://www.openbsd.org/faq/faq13.html, October 2001.

XML Specification, http://www.w3.org/TR/2000/REC-xml-20001006 , Aug 2002

XML Schema specifications, http://www.w3.org/TR/xmlschema-0, Aug 2002

XML Namespace Recommendation, http://www.w3.org/TR/REC-xml-names/

XSLT Specifications, http://www.w3.org/TR/xslt, Aug 2002 RFC2396

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Commander, Naval Security Group Command
   Naval Security Group Headquarters
   Fort Meade, Maryland

4. Deborah M. Cooper
   Deborah M. Cooper Company
   Arlington, Virginia

5. Louise Davidson
   N643, Presidential Tower 1
   Arlington, Virginia

6. William Dawson
   Community CIO Office
   Washington DC

7. Angelos Keromytis
   Dept of Computer Science
   Columbia University
   New York

8. Richard Hale
   Defense Information Systems Agency
   Falls Church, Virginia

9. Michael Green, Director
   Public Key Infrastructure Program Management Office
   National Security Agency
   Ft. Meade, Maryland

10. Cynthia E. Irvine
    Computer Science Department, Code CS/IC
    Naval Postgraduate School
    Monterey, California

11. Timothy Levin
    Computer Science Department, Code CS
    Naval Postgraduate School
    Monterey, California

12. Brig H S Lidder
    Mil Attaché
    Embassy Of India, Washington DC

13. Matt Blaze
    AT&T Labs - Research
    Florham Park, New Jersey

14. Joan Feigenbaum
    AT&T Labs – Research
    Florham Park, New Jersey

15. John Ioannidis
    AT&T Labs - Research
    Florham Park, New Jersey

16. Douglas Maughan
    DARPA / ATO
    Arlington VA

17. John Drake
    Schafer Corporation
    Arlington VA

18. Keith T. Schwalm
    White House
    Washington DC

19. David M. Wennergren
    Department of Navy
    Office of the Chief Information Officer
    Navy Pentagon
     Washington, DC

20. Elaine S. Cassara
    Branch Head, Information Assurance
    Washington, DC

21. Paul A. Karger
    Thomas J. Watson Research Center

Yorktown Heights, NY

22. Evie Spyropoulou
    Agia Paraskevi
    Athens, Greece

23. James P Anderson
    Port Washington, PA

24. George Dinolt
    Computer Science Department, Code CS
    Naval Postgraduate School
    Monterey, California